1995-09

A methodology for the integration multiple distributed heterogeneous databases: application to databases of the Tomahawk engineering community

O'Neill, Thomas Edward; Prell, Milton John

Monterey, California. Naval Postgraduate School

NAVAL POSTGRADUATE SCHOOL
MONTEREY, CALIFORNIA

THESIS

A METHODOLOGY FOR THE INTEGRATION OF MULTIPLE DISTRIBUTED HETEROGENEOUS DATABASES: APPLICATION TO DATABASES OF THE TOMAHAWK ENGINEERING COMMUNITY

by

Thomas E. O'Neill

and

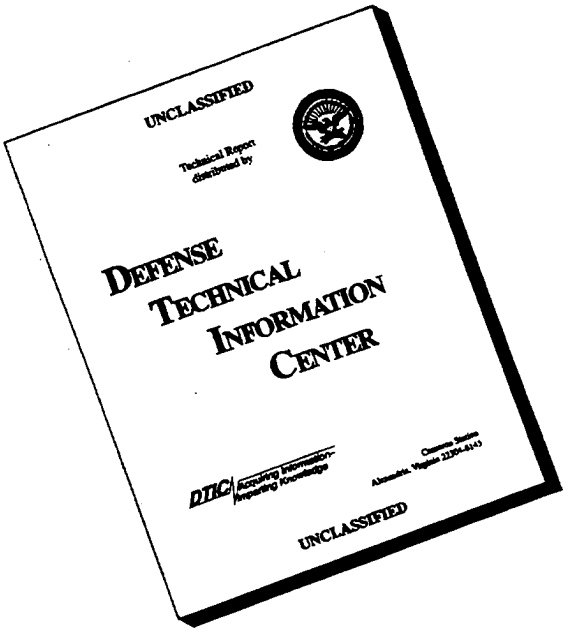
Milton J. Prell

September 1995

Thesis Advisor: Magdi N. Kamel

Approved for public release; distribution is unlimited

DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 1995	Master's Thesis

4. TITLE AND SUBTITLE
A METHODOLOGY FOR THE INTEGRATION OF MULTIPLE DISTRIBUTED HETEROGENEOUS DATABASES: APPLICATION TO DATABASES OF THE TOMAHAWK ENGINEERING COMMUNITY

5. FUNDING NUMBERS

6. AUTHOR(S)
O'Neill, Thomas E.
Prell, Milton J.

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING/ MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Port Hueneme Division, Naval Surface Warfare Center
Tomahawk Systems Engineering
4363 Missile way
Port Hueneme, CA 93043-4307

10. SPONSORING/ MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES
The views expressed in this thesis are those of the authors and do not reflect the official policy or position of the Department of Defense or the United States Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release; distribution is unlimited.

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*

Engineering 2000 is a project initiated by Naval Surface Warfare Center (NSWC) and designed to develop an infrastructure for the sharing of engineering data and information to support the current and future needs of the Tomahawk engineering support community. The goal of Engineering 2000 is to integrate the engineering, logistics, and management automated tools of the Tomahawk community into a single infrastructure. This thesis investigates the integration of Tomahawk distributed heterogeneous databases. It develops a six-step methodology for identifying an integrated strategy and architecture for heterogenous databases in a distributed environment and apply it to the two most significant databases used by the Tomahawk engineering community, Tomahawk Information Management Engineering System (TIMES) and Tomahawk Engineering Exchange Network (TEXN). The application of the methodology to these databases suggests a loosely coupled architecture for integrating their data.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Database Analysis, Database Integration, Federated Database, Data Warehouse, Entity-Relationship Diagram, Tomahawk Databases, Database Analysis, Database Integration, Conceptual Data Modeling, Schema Conflicts	178
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UL

Approved for public release; distribution is unlimited.

A METHODOLOGY FOR THE INTEGRATION OF MULTIPLE DISTRIBUTED HETEROGENEOUS DATABASES: APPLICATION TO DATABASES OF THE TOMAHAWK ENGINEERING COMMUNITY

Thomas Edward O'Neill

Lieutenant, United States Navy

B.A., University of Michigan, 1987

Milton John Prell

Lieutenant, United States Navy

B.S., Miami University, 1987

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN INFORMATION TECHNOLOGY MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL

September 1995

Authors: [redacted]

Thomas E. O'Neill

[redacted]

Milton J. Prell

Approved by: [redacted]

Magdi N. Kamel, Thesis Advisor

[redacted]

Martin J. McCaffrey, Associate Advisor

[redacted]

Reuben T. Harris, Chairman, Department of Systems Management

ABSTRACT

Engineering 2000 is a project initiated by Naval Surface Warfare Center (NSWC) and designed to develop an infrastructure for the sharing of engineering data and information to support the current and future needs of the Tomahawk engineering support community. The goal of Engineering 2000 is to integrate the engineering, logistics, and management automated tools of the Tomahawk community into a single infrastructure. This thesis investigates the integration of Tomahawk distributed heterogeneous databases. It develops a six-step methodology for identifying an integrated strategy and architecture for heterogenous databases in a distributed environment and apply it to the two most significant databases used by the Tomahawk engineering community, Tomahawk Information Management Engineering System (TIMES) and Tomahawk Engineering Exchange Network (TEXN). The application of the methodology to these databases suggests a loosely coupled architecture for integrating their data.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENT

The authors would like to acknowledge the financial support of Port Hueneme Division, Naval Surface Warfare Center. This research was part of the Engineering 2000 project initiated by the Tomahawk System Engineering office.

The support of several individuals was invaluable to the successful completion of this research. The include Cary Martinez, Derrick Moody, and Mike Grapevine at Port Hueneme Division; Kip McCullen at Weapons Control Systems-Engineering Integration Agent; and Thurman Brooks at Dahlgren Division.

The authors want to thank Professor Magdi Kamel for his guidance and patience during the performance of this research.

We would also like to thank our wives, Susan and Melissa, for their untiring support and understanding throughout this process.

I. INTRODUCTION

A. BACKGROUND

The Tomahawk Weapons System is an effective, combat proven weapon in littoral warfare. As littoral conflicts are expected to be the prevalent type of conflict in the post Cold War era, the Tomahawk Weapon System has become one of the Navy's most important weapon systems. Its significant tactical and strategic role increases the need for continuous improvement and redesign. This continuous requirement for improvement places significant demands on the engineering community responsible for its design, production, integration, support, maintenance, and use.

Responding to the demands, the Tomahawk engineering community is increasingly relying on automated tools that help employees do their jobs better, quicker, and with less resources. Information systems currently used by the Tomahawk community are, however, an amalgamation of incompatible, standalone software and hardware tools. Data cannot be exchanged between systems and most functions can only be executed in one location. The use of multiple operating systems and applications throughout the organization places heavy demands on the users. In addition, technical support and documentation is not available to facilitate integration because many of these systems were developed in an ad hoc manner, without the benefit of modern design methodologies.

In response to this situation, the Naval Surface Warfare Center Port Hueneme Division (NSWC PHD) initiated Engineering 2000, a project designed to facilitate integration and eliminate the problems caused by the operation of multiple standalone systems. The goal of the project is to integrate engineering, logistics, and management automated tools into a single environment to support the current and future needs of the Tomahawk engineering support community. (NSWC PHD, 1995)

To achieve this goal, several specific objectives have been identified. These objectives address a wide range of issues: (NSWC PHD, 1995)

- Provide a framework that enables the access to and exchange of data
- Establish standards for development and configuration management of new and existing tools
- Identify and resolve areas of duplication
- Provide a consistent infrastructure for all automated tools
- Optimize workflow processes to improve work efficiency

Engineering 2000 is divided into six phases. Phase One, the feasibility study, was conducted by Science Applications International Corporation (SAIC) and completed in November, 1994. The study included a basic analysis of tool usage, tool capabilities, tool interoperability, and system architecture. This analysis included a complete outline of the current hardware and software in use throughout the Tomahawk engineering community and recommendations for possible solutions.

Phase Two is the requirements analysis phase. It includes an analysis of the architecture, data, functions, and operations within the Tomahawk engineering community. This analysis will be more detailed and structured than that conducted in the feasibility study. It involves comprehensive review and analysis of the database, network, and other systems.

Phase Three will develop the requirements definition. In this phase, standards will be defined and hardware, interface, network, and functional requirements identified. In Phase Four, the architecture, interfaces, and functionality will be defined and a migration plan developed. System development will take place in Phase Five, with implementation of the Engineering 2000 system occurring in Phase Six. (NSWC PHD, 1995)

This thesis is part of an effort undertaken by the Naval Postgraduate School faculty and students that addresses the requirements analysis (Phase Two). Specifically, the thesis addresses the issues of identifying and resolving the overlap of functionality and data between database management systems (DBMS) used by the Tomahawk engineering community.

B. OBJECTIVES

The primary objective of our research is to develop a methodology for identifying an integrating strategy and architecture for multiple distributed heterogeneous databases within an organization.

A secondary objective is to apply this methodology to the two most significant databases used by the Tomahawk engineering community, Tomahawk Information Management Engineering System (TIMES) and Tomahawk Engineering Exchange Network (TEXN), and recommend an approach for integrating these two databases.

C. RESEARCH QUESTIONS

Several research questions were identified before conducting the research and used to focus the direction of the thesis. As we progressed, the questions were continually clarified and revised to address the actual issues of the environment under study . Some initial questions were inconsequential and thus eliminated, while new questions were developed as we progressed. The research questions address two areas: the first addresses our primary objective of developing a methodology, and the second addresses our secondary objective of applying that methodology to TEXN and TIMES. The research questions are:

Primary research question:

1. What is an appropriate methodology to use when conducting a large scale system integration across functionally diverse and geographically distributed database systems?

Secondary research questions:

2. What is the structure and functionality of the Tomahawk Engineering Exchange Network (TEXN) and Tomahawk Information Management Engineering System (TIMES) databases and how are they implemented and used?

3. Are there overlaps between TEXN and TIMES, either in their functions or their data?

4. Can the identified methodology be used to develop a strategy for combining the functions and data of TEXN and TIMES?

D. METHODOLOGY

The feasibility study conducted by SAIC was reviewed to gain a basic understanding of the databases used in the Tomahawk engineering community. Two requirements were identified: the need for more detailed information about the two databases under study and the need to develop a methodology for analyzing the databases and identifying a strategy and architecture for their integration.

An outline was developed that represented the general steps required to analyze and understand the structure and contents of the databases, identify and resolve the conflicts between them, and propose an integration strategy. This outline was revised and expanded after each step and became the basis of our methodology.

Site visits to the locations operating TEXN and TIMES were scheduled. A survey instrument was developed that covered most aspects of database design, implementation, use, and maintenance. In addition, available system documentation for TEXN and TIMES was collected.

A computer aided software engineering (CASE) tool was used to model the two databases. TEXN was modeled using a re-engineering process, while TIMES was modeled through reverse engineering. These models were then used to evaluate the structure and functionality of each database and identify the overlap between them.

Finally, potential database integration strategies were identified and evaluated for use with TEXN and TIMES. A strategy was recommended based on the evaluation of the two databases and the needs of the Tomahawk engineering community.

E. ORGANIZATION

The thesis is organized as follows. Chapter II presents a methodology for identifying an integration strategy and architecture for heterogeneous databases in a distributed environment. Chapters III and IV provide an overview of the structure and

functionality of the TEXN and TIMES databases respectively. Chapters V and VI describe the conceptual data models of TEXN and TIMES. We outline the re-engineering process used to evaluate and model TEXN and the reverse engineering process used to evaluate and model TIMES. Chapter VII discusses the overlaps between TEXN and TIMES. The semantic conflicts between the databases are assessed and all relevant redundancies examined. Chapter VIII discusses database integration strategies. Finally, Chapter IX contains recommendations, conclusions, and lessons learned.

II. METHODOLOGY FOR IDENTIFYING AN INTEGRATING STRATEGY FOR HETEROGENEOUS DATABASES IN A DISTRIBUTED ENVIRONMENT

This chapter presents a methodology for identifying an integration strategy and architecture for heterogeneous databases in a distributed environment. The remainder of the thesis applies the proposed methodology to the integration of the TEXN and TIMES databases. The methodology is a six step process for determining the best approach for integrating data from diverse, heterogeneous databases. The steps of the methodology are: 1) analyzing the databases, 2) developing a conceptual model, 3) identifying and resolving schema conflicts, 4) determining the overlap among databases and grouping them by degree of overlap, 5) enumerating alternative solutions, and 6) identifying the most appropriate solution. The flowchart in Figure 2-1 illustrates this process.

This chapter is organized as follows. Section A discusses the approach used for analyzing the existing databases. Section B discusses the importance of developing a conceptual data model for databases that require integration. It outlines methods for developing a conceptual model and describes the entity-relationship modeling technique. Section C discusses the identification and resolution of schema conflicts. Section D describes the identification of overlap across separate databases and categorizes the level of that overlap. Section E discusses alternative solutions for integrating databases based on current technologies. Finally, section F presents guidelines for recommending a database integration solution.

A. ANALYZING EXISTING DATABASES

The first step in the heterogeneous database integration process is to study the existing databases. This is accomplished by 1) collecting information through a questionnaire, 2) interviewing developers, administrators, and end-users, and 3) studying existing system documentation, screens, and reports.

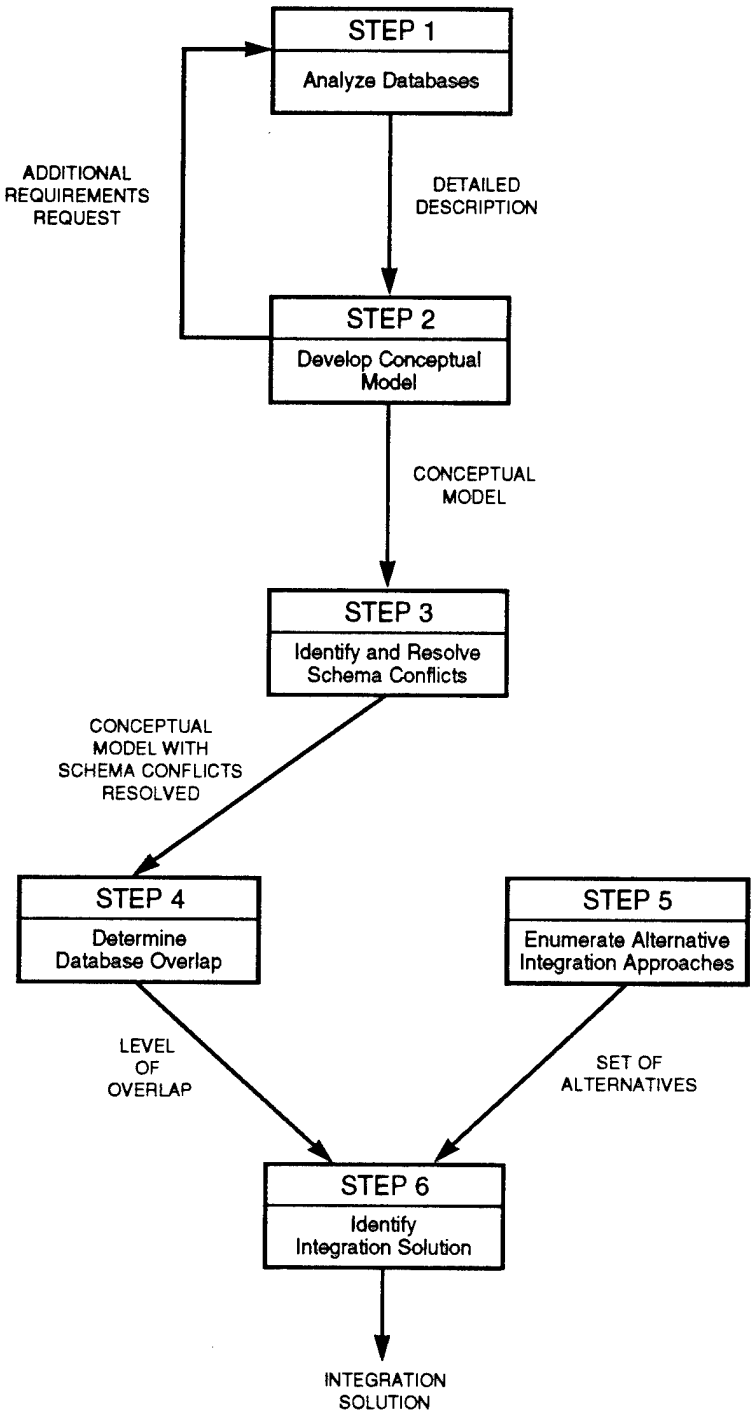


Figure 2-1. Heterogeneous Database Integration Process.

1. Data Collection Questionnaire

Preparation of the data collection questionnaire is the first step in analyzing a database. To properly perform a database analysis, it is necessary to understand the purpose, functions, components, and data structure of each database under study. A data collection questionnaire is a useful tool to structure data collection and ensure these areas are covered during an interview. A comprehensive questionnaire should cover, at a minimum, the following areas:

- Development History - Was there a preceding system? Why and when was the current database developed? What is the history of revisions to the system? What versions are currently in use?
- System Requirements - What are the hardware, software, operating system, remote access, and update time requirements?
- Volumes and Frequencies - Is system usage monitored by the operating system? How frequently is each database and each table accessed? How often is each report generated? How much disk space does each database use? What are future projections for database size? What are the performance requirements?
- Data - What is the conceptual data model, underlying model of the database, content and structure of the meta-data, data definitions, degree of normalization, and degree of data integrity?
- Applications - What applications make up the database system? What are the interfaces, views, and tools?
- Administration - Who is the database administrator? Who maintains the database? What documentation exists?
- Users - Who are the database users and what is their level of sophistication?
- Security - What are the security requirements and levels of access? Which database security application is in use?
- Backup, Recovery, and Concurrency Control - What are the back-up and recovery procedures? What concurrency control functions are employed?

The data collection questionnaire performs three key functions. First, it focuses the study on the important information for the database analysis. Second, it prepares the developers and users on the purpose of the interviews and the nature of the questions. Third, it structures the data collection, acting as a checklist to ensure complete coverage of all prepared questions.

2. Interviewing Developers and Users

The next step in conducting a database analysis is to interview the developers and users of the database. The developers of a database are intimately familiar with its structure, functions, and modifications, while users of a database understand the purposes and processes behind the data. Both perspectives are necessary to fully understand the structure, functionality, and use of a database. A face-to-face interview provides a greater depth and detail of information than any other data collection technique; responses from the subject can be immediately clarified.

Three conditions must be met for a successful personal interview: 1) availability of requested information from the respondent, 2) an understanding by the respondent of his or her role, and 3) adequate motivation by the respondent to cooperate. (Emory and Cooper, 1991) Communication with the respondent in advance of the interview is the key to successfully accomplishing these aims. Providing the respondent with the data collection questionnaire and submitting special requests before the interview will allow him or her time to gather the necessary information. Contacting the respondent and explaining the purpose and intent of the interview helps the respondent understand his or her role. Pointing out the benefits of the study and how the respondent's involvement will contribute to its success assists in motivating the respondent to cooperate. The interviewer should be aware of any threats that the study may pose to the respondent and be sympathetic to the respondent's situation.

The on-site interview provides an opportunity to gather as much data as possible, answer any questions, and establish relationships with key persons that can contribute in understanding the database. At a minimum, the interview must cover all questions on the data collection questionnaire and include a demonstration of the database system. The

demonstration should include an overview of system applications, data input and presentation screens, and output reports. Copies of design documentation, user documentation, and example reports should be obtained for reference and analysis. It is important to schedule enough time to comprehensively cover these areas. Further communications via electronic mail or telephone can answer any additional questions or requests for additional resources.

3. Documentation, Input Screen, Output Report Review

Information gathered from documentation, input screens, and output reports augments the data collected from the developer and user interviews. Documentation furnishes information on the background, applications, and operating instructions of a database. Information gleaned from input screens and output reports focuses on the usage and structure of the data. Database dictionary data is also helpful in understanding the data. This information, coupled with data collected from developer and user interviews, provides a solid foundation for the analysis of the data.

B. DEVELOPING A CONCEPTUAL DATA MODEL

A conceptual data model is an effective way for understanding the structure of a database. In addition, conceptual modeling can be used to represent, identify, and resolve the conflicts across different component databases. (Kamel, 1994) Entity-relationship modeling is an effective and commonly used conceptual modeling technique and is discussed in the following sections.

1. Entity-Relationship Model

The Entity-Relationship (E-R) diagram is a simple and widely used modeling technique for top-down database design. It is also used in a bottom-up approach for representing heterogeneous databases for the purpose of integrating their data. The E-R model consist of three basic concepts: entities, attributes, and relationships between entities.

Items of interest in the user's environment are represented by entities. Entities are real-world objects that exist independent of one another. They can be tangible objects such as ship or person, or intangible concepts like a test.

Detailed information about entities are called attributes. Attributes can be single valued, such as a hull number, or they can be multivalued or composite attributes such as an address. A key attribute uniquely identifies each entity and is used to distinguish between instances of the same entity type. Entities that do not have a unique key attribute and are uniquely identified only by their relationship to other entities are called weak entities.

Associations between types of entities are called relationships. For example, a ship is related to a test by a "has a" relationship. Relationships are characterized by their degree, cardinality ratio, and participation constraints.

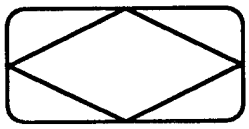
The degree of the relationship is the number of entities participating in the relationship. Most relationships are binary or of degree two. Cardinality ratio constraints refer to the number of instances of other entities in which an entity can relate, and includes one-to-one, one-to-many, and many-to-many. In a one-to-one relationship, one instance of an entity relates to only one instance of another entity. In a one-to-many relationship, an instance of one entity relates to many instances of another entity. In a many-to-many relationship, an instance of an Entity A relates to many instances of another Entity B, while an instance of Entity B relates to many instances of Entity A. A participation constraint indicates whether the existence of an entity depends on its being related to another entity through the relationship type. A ship entity that has a mandatory one-to-many "has a" binary relationship with a ship system entity is an example of a relationship and its constraints.

A superclass/subclass relationship specifies relationships in which all entities share a common set of attributes in their description, but in which the subclass entity also possesses a subset of identifying attributes. The subclass entity inherits the common attributes from its superclass entity. An "is a" relationship is an example of a superclass/subclass relationship. (Kamel, 1994) (Elmasri and Navanthe, 1989) Figure 2-2 depicts the main E-R symbols used by the data modeling CASE tool System Architect.

ENTITY

WEAK ENTITY

RELATIONSHIP



ASSOCIATIVE ENTITY

Ship - Name — ATTRIBUTE

Hull - Number [PK] — KEY ATTRIBUTE

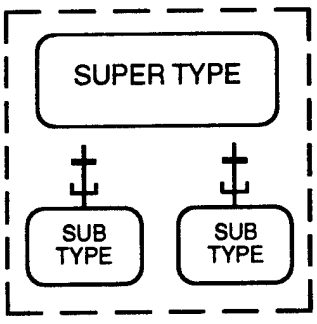
Test - Name [FK] — FOREIGN KEY ATTRIBUTE

Telephone - Number — COMPOSITE ATTRIBUTE

- area - code
- prefix
- extension

CARDINALITY RATIO OF 1 : N ,
MANDATORY PARTICIPATION

CARDINALITY RATIO OF 1 : 1 ,
OPTIONAL PARTICIPATION



A SUPER CLASS/SUB CLASS RELATIONSHIP

Figure 2-2. Entity-Relationship Model Symbols.

2. Ad Hoc Designed Databases

Ad hoc databases were not designed using a conceptual data model. In these cases, the underlying data model is usually a flat file, making data analysis difficult. Since a conceptual data model is required to represent, identify, and resolve the conflicts between the heterogeneous databases, these databases must be re-engineered, in a top-down manner, from database structure files, input forms, output reports, and other documentation. This re-engineering process includes four steps:

1. Analysis of Existing Database Structure and Reports - entails gaining an understanding of the data and identifying the data elements.

2. Identification of Entities and Attributes - involves an iterative process of generating a preliminary set of entities, assigning attributes to them, and breaking and consolidating new entities as necessary.

3. Identification of Relationships Between Entities - includes defining associations between primary entities.

4. Generation of the Entity-Relationship Diagram - involves modeling the identified entities, their attributes, and relationships in a data modeling CASE tool.

The resulting data model requires verification from the database users to ensure its accuracy.

3. Formally Designed Databases

Databases that were developed using a conceptual data model are considered formally designed. These databases are relationally structured, usually well documented, and therefore easy to study. The conceptual data model is regenerated through a reverse engineering process using a CASE tool and then compared with the original design model

to verify accuracy and identify changes that occurred since the initial design. This reverse engineering process includes four steps:

1. Reverse Engineering of Database Implementation Tables - involves using a data modeling CASE tool with reverse engineering capabilities to produce a preliminary E-R diagram from database definitions.

2. Verification of Entities and Attributes - entails verifying the entities and their attributes by comparing the preliminary model to the data dictionary and the original data model.

3. Verification of Relationships - involves verifying relationships by comparing the preliminary model to the data dictionary and the original data model.

4. Generation of Entity-Relationship Diagram - requires revising the E-R diagram as necessary to reflect the databases current structure.

The resulting data model requires verification from the database users to ensure its accuracy.

C. IDENTIFYING AND RESOLVING SCHEMA CONFLICTS

Before heterogeneous databases can be integrated, semantic conflicts between the databases must be identified and resolved. Since both databases have been modeled using the entity-relationship model, semantic conflicts can be identified by systematically comparing them against each other. We use a framework for identifying and resolving conflicts as proposed by Kamel (see Figure 2-3). (Kamel, 1994) In this approach, entities are compared against entities, attributes against attributes, and entities against attributes to identify overlaps. Semantic conflicts can be broadly categorized as schema level and data level conflicts. In this thesis, we are interested in schema conflicts only.

Schema conflicts develop at the conceptual organization and definition layer of the database. These schema level conflicts include entity level conflicts, consisting of naming, structure, and constraint conflicts; attribute level conflicts, containing name, structure, and

Schema Level Conflicts

1. *Entity Level Conflicts*

   Naming conflicts
   - Synonyms (Same real world entity named differently in different databases)
   - Homonyms (Different real world entities have the same name in different databases)

   Structure conflicts
   - Missing attributes
   - Overlapping attributes

   Constraint conflicts
   - Different cardinalities
   - Different participation

2. *Attribute Level Conflicts*

   Naming conflicts
   - Synonyms (Same real world attributes named differently in different databases)
   - Homonyms (Different real world attributes have the same name in different databases)

   Structure conflicts
   - Atomic vs. composite attributes

   Constraint conflicts
   - Type/Length clash (Equivalent real world attributes have different data type/length definitions in different databases)
   - Range clash (Equivalent real world attributes have different allowable range definitions in different databases)

3. *Entity Attribute Level Conflicts*
   - Different representation conflicts (Equivalent information is represented as an attribute of an entity in one database and as either a separate entity or attribute(s) of a Generalization/Specialization entity structure in another)

Data Level Conflicts

1. *Inconsistencies*
   - Different values returned by different databases for the same fact

2. *Data Representation Conflicts*
   - Dissimilar Expressions (Different expressions are used by different databases to describe the same fact)
   - Dissimilar Units (Different units of measurement are used by different databases to represent values for the same attributes)
   - Dissimilar Precisions (Different precisions are used in different databases to represent values for the same attributes)

Figure 2-3. Framework For Semantic Heterogeneity From (Kamel, 1994).

constraint conflicts; and entity attribute level conflicts which involve attributes in one database being represented as entities in another.

Synonyms and homonyms are examples of naming conflicts, at both the entity and attribute level. Synonyms occur when identical entities in two different databases have equivalent entities with differing names. Homonyms occur when dissimilar entities in two different databases have the same name. The same definitions of synonym and homonym apply at the attribute level.

Entity structure conflicts occur when equivalent entities have overlapping or incomplete attribute sets. Entity constraint conflicts are caused by dissimilar relationship cardinalities or by differing participation requirements for equivalent relationships between two schemas.

Attribute structure conflicts occur when information in one database is represented atomically while in another database it is represented as either separate attributes or a composite attribute. Attribute constraint conflicts are caused by differing detailed descriptions of an attribute and are categorized as type clashes, length clashes, or range clashes. Type clashes occur when the types of equivalent attributes differ, for example, character as opposed to numeric. Length clashes occur when equivalent attributes have differing numbers of characters in a field. Range clashes occur when equivalent attributes have differing allowable sets of values. (Kamel, 1994)

D. DETERMINING DATABASE OVERLAP

The first step involves determining the overlap between the databases and grouping the databases by degree of overlap. The degree of overlap can be determined by identifying redundancies at the conceptual level. Identical real-world objects common in heterogeneous databases constitute database overlap. These objects can be entities or attributes, and are identified through the careful examination and comparison of data definitions. After all overlapping objects have been identified, the percentage of the

overlapping objects between databases is calculated. The overlap is then grouped into three categories:

- Minimal overlap - databases that have less than 10 percent overlap
- Moderate overlap - databases that have between 10 and 50 percent overlap
- Strong overlap - databases that have greater than 50 percent overlap

E. DATABASE INTEGRATION ALTERNATIVES

This step entails enumerating solutions for integrating the databases based on the current technology. After a full analysis of the databases, alternative integration approaches can then be evaluated for applicability. Those approaches that may be applicable can be chosen as a set of integration alternatives.

We examined four alternatives: full merging, tightly coupled federated databases, loosely coupled federated databases, and data warehousing. In the full merging approach, all heterogeneous databases are merged into a single unified database. In the tightly coupled federated database approach, each independent database becomes part of a federation of databases through a global schema that represents the integration of the component local schemas. In a loosely coupled federated database approach, each independent database becomes part of the federation of databases without a global schema; the databases are accessed using a common language or a gateway. A data warehouse integrates data from heterogeneous operational databases into a separate "warehouse" database for the purpose of management analysis and decision support.

F. INTEGRATION RECOMMENDATION

The final step is to identify the most appropriate solution for the integration of the databases. Based on the overlap analysis and the pros and cons of the alternative integration approaches, the solution that best supports organizational requirements and constraints should be chosen. We provide the following general guidelines.

For integration of databases with strong overlap (greater than 50 percent), we recommend a full merging of the databases. Since there is so much overlap, merging the database would eliminate enough redundancy to warrant a full redesign of the database.

For integration of databases with moderate overlap (between 10 to 50 percent), we recommend a tightly integrated federated database approach. In this situation, database interaction is frequent enough to warrant the development of a global schema, but not enough for a full redesign.

For databases with minimal overlap (less than 10 percent) we recommend a loosely federated database integration approach. Occasional database interaction does not require a seamless interface, only the ability to link the data.

If the purpose of the database integration is to provide data for management decision support systems, then we recommend the implementation of a data warehouse. In this case, decision support data is time variant and non-volatile; it does not require data accuracy up to the time of access.

In the remainder of the thesis we apply the proposed methodology to two of the Tomahawk community databases, TEXN and TIMES.

III. TOMAHAWK ENGINEERING EXCHANGE NETWORK (TEXN) OVERVIEW

This chapter presents an overview of the Tomahawk Engineering Exchange Network (TEXN). The information for this overview was gathered through an interview with the TEXN database administrator using a survey questionnaire. This questionnaire is given in Appendix A.

This chapter is organized as follows. Section A reviews the history of TEXN, including a brief description of its development, planned changes, and versions. Section B discusses TEXN's system requirements, including hardware, software, and remote access. Section C discusses TEXN's data volume and the frequency of its access. Section D briefly examines the data structure aspects of TEXN, including conceptual model, data dictionary, database size, normalization, data redundancy, and data integrity. Section E reviews the TEXN applications, reports, DBMS, user interface, levels of access, data storage, and query language. Section F describes the users and database administration. Section G examines security functions and levels of access. Finally section H covers backup and recovery.

A. DEVELOPMENT HISTORY

1. Development

In 1990, Weapon Control Systems-Systems Engineering Integration Agent (WCS-SEIA), then Tomahawk Weapon Systems-Systems Engineering Integration Agent (TWS-SEIA), developed the Tomahawk Engineering Exchange Network (TEXN), an unclassified engineering management tool, to enhance the communication and information exchange between the ten agencies that support the Tomahawk Weapons System. TEXN was initiated to respond to the Tomahawk community's need for a readily accessible, accurate, reliable, and unclassified engineering management tool that provided electronic mail data interchange between agencies and real time online database access (SAIC, 1994). Before TEXN's development, no automated engineering tool was available, and the functions TEXN currently performs were handled manually.

TEXN, developed using the Microsoft database management software FOXPRO (FOXBASE for Macintosh), consists of ten applications accessible locally through WCS-SEIA's local area network, or remotely via multiple phone line connections. These database applications are linked through a common user interface but do not share data.

Since its development, TEXN has undergone one major modification, the elimination of its Government Furnished Equipment (GFE) and Ship Class/Hull Information (SHP) modules. This change to TEXN is not documented.

2. Planned Changes

WCS-SEIA has planned short and long term changes to TEXN. Three short term enhancements are scheduled. First, the database management system will be upgraded to FOXPRO 2.5 for both IBM compatible and Macintosh platforms, allowing these systems to share a common database. Second, the Government Furnished Equipment (GFE) and Ship Information (SHP) applications will be refurbished and reinstituted. Third, the Test Problem Reports (TPR) and Action Items (AI) applications will be ported to a standalone environment with upload and download capabilities.

Three long term changes are planned for TEXN. First, TEXN will be ported to FOXPRO 2.3 for UNIX which will then allow the common database to serve all three operating systems. Next, TELNET access will be incorporated to access TEXN remotely using the INTERNET. Finally, distributed site databases will be implemented.

3. TEXN Versions

There is only one version of TEXN in use. The Macintosh version was written independently using FOXBASE, rather than ported from the DOS version of FOXPRO.

B. SYSTEM REQUIREMENTS

1. Hardware Requirements

The TEXN architecture consists of a local area network (LAN) of personal computers that access a file server which stores the TEXN applications, DBMS, and

database. A multi-function communications server provides remote access to the file server (see Figure 3-1).

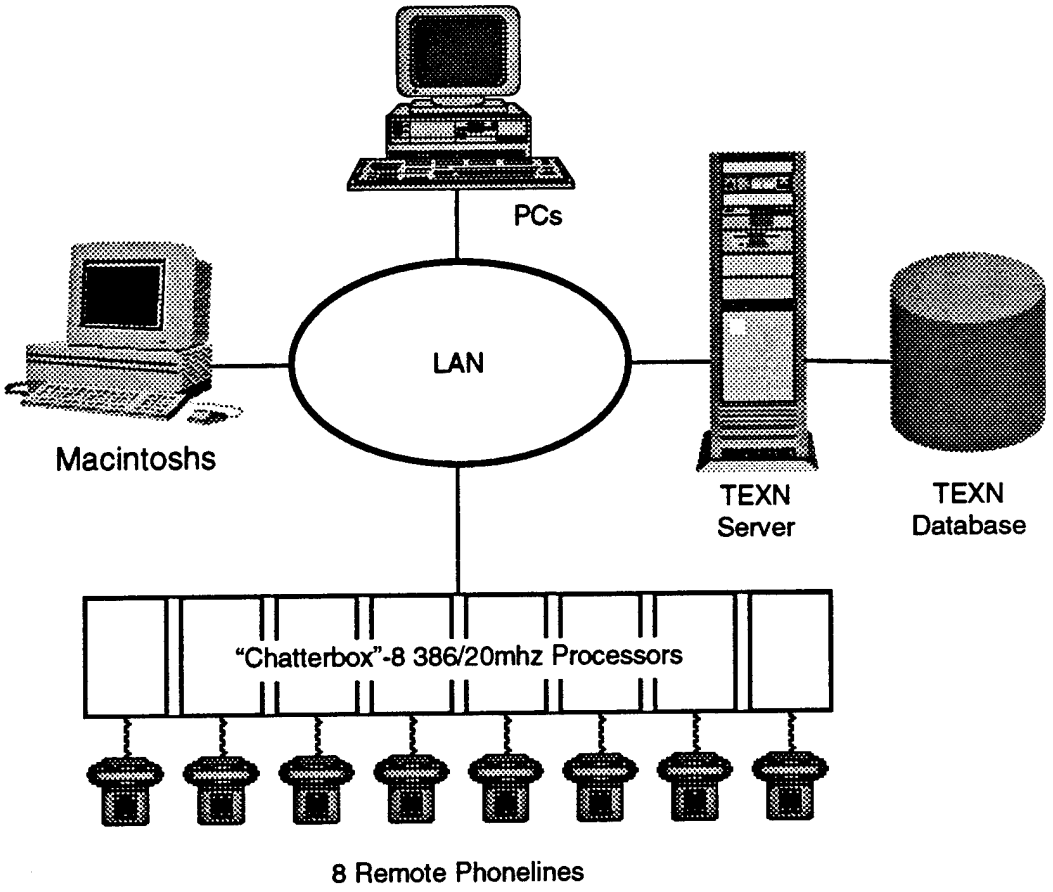


Figure 3-1. TEXN Server Configuration.

TEXN applications run on an IBM compatible or Macintosh environment. Specifically, the minimum platform requirement is an IBM XT compatible or a Macintosh Plus personal computer. A modem is required for remote access.

The TEXN file server is a 486 level IBM compatible. TEXN accommodates remote access through a multi-function communications server, called a chatterbox, which contains eight 386/20 MHZ processors with 8 MB of RAM and is connected to eight

outside lines. The remote stations dial in and capture one of the processors which upload the TEXN application and communications software.

The installed hardware is sufficient to handle the current TEXN traffic. The configuration is restricted by the eight processors in the chatterbox, its associated outside lines, and the 9600 bps speed of the modems.

2. Software Requirements

TEXN uses a single, integrated, commercial-off-the-shelf database management system: Microsoft FOXPRO for the IBM compatible platforms and FOXBASE for Macintosh platforms. It runs on the following operating systems: DOS 3.1 (or later), Windows 3.1 (or later), or Macintosh System 6.07 (or later).

3. Remote Access

The purpose of TEXN is to provide real-time database access and electronic data interchange between agencies; therefore remote access is a present and future requirement.

C. VOLUMES AND FREQUENCIES

1. Volumes

The storage size of the TEXN application, DBMS, and database on the server is 411 MB. The server reserves 307 MB of this storage space for the TEXN user database. A TEXN application uses up to 1 MB of RAM during its execution on a client computer.

2. Frequency Metrics

No current usage metrics for the TEXN system were available; however the system administrator indicates that Action Item and Calendar of Events are the most frequently accessed applications.

D. DATA

1. Conceptual Model

TEXN was developed in an ad hoc manner; therefore no conceptual model of the database was developed. The database uses a flat file design. Each application maintains its own set of tables. An example of such an application is Test Problem Reporting, which maintains a TPR and other associated tables.

2. Data Definitions

The structure of each table and associated attributes are presented in Appendix C.

3. Database Size

TEXN is a relatively small database with approximately 50 tables and 300 attributes, including the archival databases.

4. Normalization

TEXN uses a flat file architecture and is, therefore, not normalized.

5. Data Redundancy

TEXN maintains one database, located on the TEXN server, for both the PC and Macintosh user base. While some databases will strategically use data redundancy to improve application performance, TEXN does not employ this approach.

6. Data Integrity

A multi-user environment presents a risk of violating data integrity. TEXN runs on a multi-user database management system that provides concurrency control, maintaining data integrity through locking mechanisms that prevent simultaneous updates to the same data.

E. APPLICATIONS

1. Application Programs

TEXN is a suite of ten loosely coupled applications, through a menu interface, that provide information sharing within the Tomahawk community. It includes Action Items, Network Trouble Reporting, Calendar of Events, Point of Contact, Cable Run Sheets, Test Problem Reporting, Fleet Problem Reporting, External Problem Reporting, and administrative support functions. These applications are written using FOXPRO's proprietary language to access the related database.

a. Action Items

The Action Item module tracks the status of official actions assigned by one Tomahawk agency to another Tomahawk agency. Options include adding new action items, editing existing action items, viewing the database, searching the database for specific items, and generating reports. The Action Item Module accesses two databases: active and archive. The active database contains action items that are current, delinquent, and closed for less than one month. The archive database includes action items closed for more than one month. The archive database cannot be updated.

b. Network Trouble Reporting

The Network Trouble Reporting module reports errors encountered in TEXN and takes user requests for system enhancements. Options include adding new network trouble reports (NTRs), editing existing NTRs, viewing the database, searching the database for specific entries, and generating reports.

c. Calendar of Events

The Calendar of Events module schedules and tracks official Tomahawk agency events. Options include adding new events, editing existing events, viewing the database, searching for specific events, and generating reports. Calendar updates are automatically sent to TEXN users every Monday.

d. Point of Contact

The Point of Contact module contains an online phone and mail directory for Tomahawk agency personnel. Options include adding new organizations and personnel, editing existing points of contact (POC), searching the database for specific POCs, and generating reports.

e. Cable Run Sheets

The Cable Run Sheet module maintains a database of cable information, by hull, of all Tomahawk ships. The module displays Tomahawk related installation drawing numbers, cable data (in header and pin termination form), and connector backshell information. Options include adding new cable information, editing existing cable information, and viewing the database. The cable program uses two databases: one with the cable header information and the other with cable termination information.

f. Test Problem Reporting

The Test Problem Reporting module maintains reports of Tomahawk Weapon System installation and checkout (INCO) problems for each new suite that is installed and made ready for custody transfer to the fleet. Options include adding new problems, editing existing problems, viewing the database, searching the database for specific problems, and generating reports. This module accesses two databases: active and archive.

g. Fleet Problem Reporting

The Fleet Problem Reporting module maintains reports of Tomahawk Weapon System problems from fleet units and NAVSEA agents. Options include adding new problems, editing existing problems, viewing the database, searching the database for specific problems, and generating reports. Problems are submitted online or retrieved via message traffic. This module contains two databases: Fleet Problem Reports and a buffer. The buffer database withholds new problems from public view until the database manager has screened and reviewed the problem and transferred it to Fleet Problem Reports database for public view.

h. External Problem Reporting

The External Problem Reporting module maintains reports of Tomahawk Weapon System test problems discovered by non-Tomahawk test agencies.

i. Administrative

TEXN also supports administrative functions of password changes, file transfers, and electronic mail.

2. Reports

The TEXN database produces the following reports:

- INCO Anomaly Report
- Test Problem Report
- Fleet Problem Report
- External Test Report
- Network Trouble Report
- Monthly Calendar

3. User Interface

TEXN implements a menu driven user interface that links all program applications through a single screen. The applications screens are DOS based entry, edit, and view screens with pop-up help and selection lists. Figure 3-2 presents a sample entry screen.

4. Query Language

The database application uses the Microsoft proprietary query language. It does not support SQL.

F. USERS

1. Primary Users

As of March 1994, 377 individuals use TEXN at multiple sites in 13 different states. All users are in the Tomahawk engineering support community. We characterize them as casual users that occasionally access TEXN for needed information.

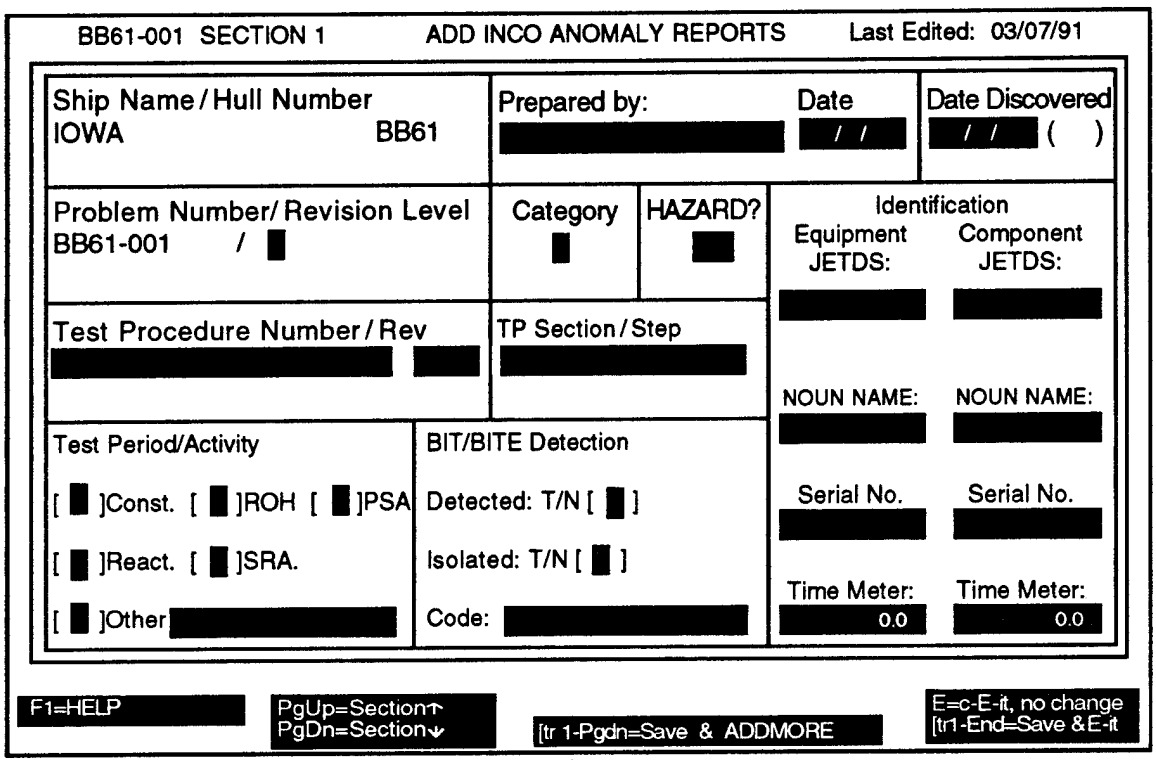


Figure 3-2. User Input Screen

2. Database Administration

WCS-SEIA administers the central TEXN database in St. Louis, MO. Each remote site appoints a local TEXN representative to manage TEXN client programs and to liaise with WCS-SEIA. A single person, Kip McCullen, maintains the TEXN database and database applications. He has been trained in and has a thorough knowledge of TEXN.

G. SECURITY

1. Security Functions

Front end security is provided through Novell local area network (LAN) software. This security includes user account and password protection with varying levels of privileges based on user needs and trust. TEXN is a completely unclassified system. No separate database security program is used.

2. Levels of Access

TEXN provides multiple levels of user access. Depending on their privileges, users can initiate, edit, or view data within the TEXN applications. Specifically, three levels of access exist within TEXN. Level 1 users can add new passwords, modify passwords, modify access levels, and change any data on the screen. Level 2 users can change their own password, edit some data on the screen, and read all of the data. Level 3 users can change their own password and read data.

H. BACKUP AND RECOVERY

The TEXN system administrator performs daily full system backups. The most data that can be lost is one day's worth of updated data. When the database crashes, the most recent system backup is loaded. Since data updates are infrequent, the impact of the lost data is minimal.

In the next chapter we will examine the Tomahawk Information Management Engineering System (TIMES).

IV. TOMAHAWK INFORMATION MANAGEMENT ENGINEERING SYSTEM (TIMES) OVERVIEW

This chapter presents an overview of the Tomahawk Information Management Engineering System (TIMES). The information for this overview was gathered during an interview with the TIMES database administrator using a survey questionnaire and from TIMES design and user documentation. The questionnaire is given in Appendix A.

This chapter is organized as follows. Section A reviews the history of TIMES, including a brief description of its development, planned changes, and versions. Section B discusses TIMES system requirements, including hardware, software, and remote access. Section C discusses TIMES data volume and the frequency of its access. Section D examines the data structure aspects of TIMES, including conceptual model, data dictionary, database size, normalization, data redundancy, and data integrity. Section E reviews the TIMES applications, reports, DBMS, user interface, levels of access, data storage, and query language. Section F describes the users and database administration. Section G examines security functions and levels of access. Finally, section H covers backup and recovery.

A. DEVELOPMENT HISTORY

1. Development

The Tomahawk Information Management Engineering System (TIMES) is a database system used to provide and track configuration management information for the Tomahawk Weapon System. This includes The Tomahawk Weapon Control System (TWCS), Mission Planning (MP), and other Strike System programs. (NSWC DD, Mar 1993)

TIMES replaced the TWCS Status Accounting Database, a system developed using Data Retrieval System (DRS) and running on a VAX 8350. (NSWC DD, Mar 1993) This system was in use for eight years. Additional Mission Planning requirements were handled using a temporary database built with the microcomputer-based Paradox database management system. This system used a simple, flat file database

structure. The DRS system was slow and antiquated, and the Paradox system was only intended as a temporary fix. These factors, combined with emerging requirements and the need for additional functionality, led to the development of TIMES.

Development of TIMES began in April of 1992. (NSWC DD, Mar 1993) Most of the work was performed at NSWC Dahlgren Division (NSWC DD), with significant design and development support from an outside contractor, INCASE. (SAIC, 1994) Most of the design work was done using Oracle CASE tool, and a complete entity-relationship diagram was generated. The design was intended to meet the functionality requirements of the old system, while moving from a hierarchical structure to a relational data structure. No significant new functionality was added, and little user input was taken prior to determining the new design. The lack of user input was partially due to low response rates when user input was requested. (SAIC, 1994)

2. Planned Changes

TIMES was intended to support all Strike System programs, including both Tomahawk and Unmanned Air Vehicles (UAV) and to be developed using a phased development approach. The initial design strategy included long range plans for an integrated information management system. Four successive baselines were recommended, with the first intended to handle configuration management. (NSWC DD, Mar 1993)

Baseline 1 supports configuration status accounting and is the only baseline developed to date. No other development has been done under the original strategy. Some work has been conducted to upgrade TIMES to a more user friendly environment. This project is tentatively known as TIMES Light, although it is not an official project and has no permanently assigned personnel. This work primarily revolves around implementing a Graphical User Interface (GUI), but does not further the original design goal of an integrated information system.

There is a plan to enhance remote access capabilities. The system now uses modems for dial-in access at 9600 bps. The modems are being upgraded to faster speeds.

TIMES is implemented using Oracle 7.0 as its relational database engine. There are plans to upgrade to Oracle 8.0 but they are not specific. No steps have been initiated to investigate the process involved for the upgrade.

3. TIMES Versions

TIMES 1.3 is the current version and is the only one in use. There was no documentation available beyond version 1.0. Several new tables were added and modified since the original version, but these changes are undocumented. We do not know if any functional or interface changes have been made since version 1.0.

B. SYSTEM REQUIREMENTS

1. Hardware Requirements

TIMES application and data reside on a Sun SPARC 10 server which is connected to other workstations and terminals through an Ethernet LAN. Access to TIMES requires a VT100 terminal, DEC VT terminal, X-terminal, or Sun workstation. (NSWC DD, Jul 1993) Most of the on-site use in Dahlgren is through terminal emulation, using desktop 486 level PCs connected to the network.

Remote users access the system using direct modem connections. There are no dedicated high speed lines, and all of the access is via low speed modems (9600 bps or less). There is access available via MILNET, but it also appears to come over low speed modem lines.

Printed output is handled by LaserJet 2.0 and above, HPGL, and Postscript printers. Since all printer types are not supported by TIMES, users must sometimes transfer a file to another machine in order to print it. (SAIC, 1994)

The Ethernet LAN is sufficient to handle local traffic at Dahlgren. The slow speed of remote access is considered inadequate for remote users.

2. Software Requirements

TIMES uses an integrated, commercial-off-the-shelf (COTS) database management system: Oracle 7.0, Oracle Terminal 1.0, and Oracle Data Query 3.1. The operating system used is SunOS 4.1.3. An X-terminal, VT100, or DEC VT emulator is required for all PCs used to access the database. There is no restriction on the type of operating system used by the PCs.

3. Remote Access

TIMES is designed to provide real-time database access to all agencies involved in Tomahawk configuration management. The current remote access requirements are not being met and the need for remote access is expected to increase. A batch function for submission of data has been proposed because of the current slow remote processing. This is not considered to be necessary because the problem is not with processing speed but with access speed. (SAIC, 1994)

C. VOLUMES AND FREQUENCIES

1. Volumes

There are 100 megabytes (MB) of disk space set aside for data, with about 40 MB of the space currently in use. The database contains approximately 20,000 system trouble reports (STR), with each STR making up an individual record. Most of these are archival in nature, meaning that they are no longer open or under investigation. No work has been done to determine future needs for storage space.

2. Frequency Metrics

Frequencies of use are not tracked and the database administrator has no knowledge of any system capabilities to monitor usage. TIMES currently produces approximately 40 different reports, but there is no tracking of the reports to determine which are used most often and by whom.

D. DATA

1. Conceptual Model

TIMES was developed using Oracle CASE and is a true relational database. The initial design documents used as the inputs to Oracle CASE were not available. The entity-relationship diagram of Baseline 1 is documented and available. (NSWC DD, Apr 1993)

2. Data Definitions

The data dictionary is thorough and breaks the data down by each individual table. (NSWC DD, Jul 1993) The complete structure of each table is outlined and includes information such as type, whether the field is required or optional, size, and description. Primary and foreign keys are also listed for each table. The only potentially important information missing is a description of minimum or maximum values, but these do not exist for most fields.

The data was defined based on the previous system data definitions. Some user input was taken to clarify and enhance the data. There was no effort made to conform to the common data standards now being worked on for DoD.

3. Database Size

TIMES is relatively small, with only about 40 MB of data currently on the system. There are approximately 50 tables in the database; about 8 of those are simple look-up tables.

4. Normalization

Since TIMES is a relational database, normalization is an important concern. There is no normalization information in the documentation, but the database administrator believes TIMES is in at least the third normal form.

5. Data Redundancy

There is no indication of controlled redundancies within TIMES.

6. Data Integrity

Oracle 7 is a robust DBMS that is in use throughout industry and government. It provides very capable integrity controls, using locking features that prevent simultaneous updates to the same data.

E. APPLICATIONS

1. Application Programs

TIMES is a single function application. It handles status accounting and provides some utility functions such as e-mail, printing, and file editing.

2. Reports

There are a number of standard reports generated by the system, but none of them require any special applications. There is also an extensive capability for generating custom reports based on data queries.

3. User Interface

The interface for the system is menu-driven and forms based. While intended for relatively naive users, the interface is character based and can be cryptic and confusing to users not familiar with the system.

4. Query Language

Oracle 7 is a SQL based DBMS. It does support SQL queries, although the users interface with the system through forms and thus are not required to learn and know SQL.

F. USERS

1. Primary Users

As of November 1994, approximately 340 individuals use TIMES at about 18 different sites. All users are in the Tomahawk engineering support community. We characterize most of these users as casual; they do not need to update data in the system

very often. There is a core of experienced users, primarily at Dahlgren, who use this system routinely.

2. Database Administration & Maintenance

NSWC Dahlgren is the administrating entity for TIMES. Thurman Brooks is the database administrator, with assistance from Cornell Downs, an employee of Vitro Corporation. Each remote site has a security point of contact (SPOC) who handles the obtaining of accounts for remote users. (NSWC DD, Jul 1993) Cornell Downs handles all system maintenance as part of the contractor support for TIMES.

3. User Documentation

There is a users manual that gives an overview of TIMES' functions. It includes explanations of the query functions, definitions of many system terms, and diagrams of several screens. It does not include a tutorial walk-through. There is also a TIMES help desk that supports communication, hardware, software, and the TIMES database itself. (NSWC DD, Jul 1993) The help desk function appears to be a collateral duty for the Vitro representative.

G. SECURITY

1. Security Functions

Access to TIMES is controlled through a password system. Once in the system, users are responsible for changing their own passwords periodically. TIMES is a completely unclassified system. No separate database security program is used.

2. Levels of Access

TIMES provides multiple levels of user access. The database administrator assigns the various levels of access to each user. There is no documentation about the different levels.

H. BACKUP AND RECOVERY

The TIMES system administrator performs nightly incremental backups to tape. A full backup is done monthly. The most that could be lost due to a system problem is one day worth of data updates. The backup tapes are all stored next to the TIMES server. This could cause a serious problem in the event of a fire or similar catastrophe.

In the next chapter, we will discuss the TEXN conceptual data model.

V. TEXN CONCEPTUAL DATA MODEL

This chapter describes the conceptual data model for TEXN. It is organized as follows. Section A provides the motivation for developing a conceptual data model for TEXN. Section B describes the process of generating the TEXN entity-relationship model by analyzing the existing TEXN database structure and reports to identify the entities, their attributes, and the relationships between them. Section C examines TEXN entities, listing their attributes and relationships. Finally, section D briefly examines the data and functional overlaps within TEXN.

A. BACKGROUND

Since TEXN was developed without the benefit of a formal design methodology, the resulting data structure is a collection of unnormalized flat files with no defined relationships. As with any unnormalized data, there is the risk of modification anomalies; problems caused by the actions of insertion, deletion, or modification of data in a data table. This risk can include data loss, data duplication, or data staleness (not updated).

To improve our understanding of the data and its relationships in TEXN, we decided to develop a high level conceptual data model based on the existing data structures, input screens, and output reports. A conceptual model provides a simplified and graphical view of the real world objects, their properties, and relationships to help designers and users to better communicate and understand the data.

B. DEVELOPING THE TEXN ENTITY-RELATIONSHIP MODEL

Because a conceptual model of TEXN did not exist, it was necessary to re-engineer an E-R model from the FOXPRO database structure summaries and sample TEXN input screens and reports. The summaries included a listing of data element names, format, and length. These database structure summaries were grouped by TEXN modules. The data elements were compared to the sample TEXN reports to better understand the purpose of the data elements. Next, we identified entities, their attributes, and their relationships. Finally, we generated the entity-relationship diagram.

1. Analysis of Existing Database Structure and Reports

To gain an understanding of the data, we first studied the FOXPRO database structure summaries. These database summaries organized data into flat file tables, one for each TEXN application, with duplicate tables for archived data. The tables listed data elements by name, type, and length. The names of each data element were usually cryptic and limited to ten characters, a FOXPRO limitation. In many cases it was difficult to understand their semantics or purpose, due to the name length limitations.

We then compared the FOXPRO database structure summaries to the sample TEXN input screens and reports to better understand the meaning and purpose of each data element. The majority of the elements were identified and discerned. However, some data elements could not be identified and consultation with the designers was designers to understand their semantic information. Some items did not belong in the conceptual data model and were discarded.

2. Identifying Entities and Attributes

Having studied the database structure summaries and the TEXN screens and reports, our next step was to develop the entities and their attributes. This is an iterative process of generating a preliminary set of entities, assigning attributes to them, and then breaking and consolidating these entities as new data is analyzed.

Since the data was already grouped by TEXN application, we used these groupings as the beginning selection of entities. The starting eight entities included: ACTION_ITEM, NETWORK_TROUBLE, EVENT, POINT_OF_CONTACT, CABLE_RUN_SHEETS, TEST_PROBLEM, FLEET_PROBLEM, and EXTERNAL_PROBLEM.

Investigation of Test Problem Reports revealed that this application actually produced two separate reports: Test Problem Reports and INCO Anomaly Reports. These reports contained some distinctly different attributes, so we created two separate entities named TEST_PROBLEM and INCO_ANOMALY_PROBLEM.

Studying the different trouble reports, we concluded that Test Problems, INCO Anomaly Problems, and External Problems were generated from a test of the ship's Tomahawk Weapon System (TWS). Tests of a TWS by the Tomahawk engineering support community generates a problem that is reported through either a Test Problem Report or INCO Anomaly Report. A test conducted by any agency outside the Tomahawk engineering support community produces a problem that is reported through an External Problem Report. This was reflected in the E-R model by creating the entity EXTERNAL_PROBLEM. An instance of a test on specific ship at a specific time produces a problem. We created entities of TEST and SHIP to represent this situation.

A problem reported from the fleet that was not discovered through specific TWS test is reported through a Fleet Problem Report. An instance of a problem with a TWS on a specific ship generates a Fleet Problem. To represent this concept, we created the entity SYSTEM.

For each of the entities, we listed their relevant attributes, determined their keys, and then identified common attribute classes between the entities. These overlapping attributes were made into separate entities, and the foreign keys appropriately propagated. Specific examples of this process follow.

We found the attribute class of persons in all entities except CABLE_RUN_SHEETS and made these person objects into a separate entity called PERSON, transferred the attributes in the POINT_OF_CONTACT entity to the new entity PERSON, and eliminated the POINT_OF_CONTACT entity . We then designated the person attributes in the other entities as foreign keys.

Discovering that TEST_PROBLEM, FLEET_PROBLEM, EXTERNAL_PROBLEM, and INCO_ANOMALY all shared common attributes, we extracted these attributes from the individual entities and made them into a separate PROBLEM supertype entity with the Problem-Number as its primary key. The format of Problem-Number exactly matched the primary keys of TEST_PROBLEM, INCO_ANOMALY, and EXTERNAL_PROBLEM; therefore we selected Problem-Number as both a primary and foreign key for these three entities.

Organization attributes resided directly in PERSON, FLEET_PROBLEM, and INCO_ANOMALY. We created a separate entity called ORGANIZATION and made the organization attributes in other entities into foreign keys.

CABLE_RUN_SHEETS contained information on ships wiring diagrams and cables associated with a TWS. Further investigation of this application revealed that a system contains many instances of units, a unit contains many instances of cables, a unit is associated with many instances of systems, and a cable is associated with many instances of units. We split the entity CABLE_RUN_SHEET into entities UNIT and CABLE, and shifted the attributes of the four types of ship's drawings to a new weak entity named DIAGRAMS, with Hull-Number, from the entity SHIP, as its primary key.

3. Identifying the Relationships

Having established the primary entities, we next examined the relationships between these entities. A foreign key must be associated with every relation. Entities with many-to-many relationships must be resolved to two one-to-many relationships through an intersection entity.

We discovered four many-to-many relationships that required resolution: SHIP and SYSTEM, SHIP and TEST, SYSTEM and UNIT, UNIT and CABLE. These relationships required the creation of SHIP_SYSTEM, SHIP_TEST, SYSTEM_UNIT, and UNIT_CABLE entities. The identification of all our entities for the E-R diagram was complete.

4. Generating the Entity-Relationship Diagram

With the entities identified, their attributes and keys defined, and their relations determined, the final step in the modeling process is to generate an entity-relationship diagram. We used the CASE tool System Architect 3.0 to graphically represent the E-R diagram.

The entities in System Architect E-R diagrams include attributes, which decompose into data structures or data elements. Data structures consist of other data structures or data

elements. Data elements, the lowest decomposed level of data, are defined by data domains, e.g., text or numeric.

In describing the re-engineered entity-relationship model of TEXN, the entity will be defined, its attributes listed, primary and foreign keys identified, and its relationships described. The data description of the entity relationship model follows.

C. ENTITY DESCRIPTIONS

1. SHIP

The entity SHIP describes a Tomahawk Weapon System platform. It is identified by attribute Hull-Number and includes Ship-Name.

SHIP has an optional one-to-many "has a" relationship with SHIP_TEST, a mandatory "has a" relationship with DIAGRAM, and a mandatory "has a" relationship with SHIP_SYSTEM.

2. TEST

The entity TEST describes a test or inspection performed on a Tomahawk Weapon System. It is identified by the attribute Test-Name and includes Test-Description.

TEST has an optional one-to-many "has a" relationship with SHIP_TEST.

3. SHIP_TEST

The entity SHIP_TEST describes a specific test held on a specific ship. It is identified by the composite attributes of Hull-Number, Test-Name, and Test-Date, and includes Period.

SHIP_TEST has optional many-to-one relationships with TEST and SHIP, and an optional one-to-many relationship with PROBLEM.

4. SYSTEM

The entity SYSTEM describes a generic Tomahawk Weapon System. It is identified by JETDS and includes Noun-Name.

SYSTEM has an optional one-to-many "has a" relationship with SHIP_SYSTEM and a mandatory "has a" relationship with SYSTEM_UNIT.

5. SHIP_SYSTEM

The entity SHIP_SYSTEM describes a specific Tomahawk Weapon System on a specific ship. It is identified by JETDS and Hull-Number and includes Serial-Number.

SHIP_SYSTEM has an optional one-to-many "has a" relationship with PROBLEM, a mandatory many-to-one "has a" relationship with SHIP, and an optional many-to-one "has a" relationship with SYSTEM.

6. DIAGRAM

The weak entity DIAGRAM describes a Tomahawk Weapon System's diagrams for a particular platform. It is identified by Hull-Number and includes AC-Power-Distribution, DC-Power-Distribution, TWCS-Block-Diagram, and TWCS-Cable-List.

DIAGRAM has a mandatory one-to-one "has a" relationship with SHIP.

7. UNIT

The entity UNIT describes a generic unit of a Tomahawk Weapon System. It is identified by Unit-Number.

UNIT has optional one-to-many "has a" relationships with SYSTEM_UNIT and UNIT_CABLE.

8. SYSTEM_UNIT

The entity SYSTEM_UNIT describes a specific unit of a Tomahawk Weapon System. It is identified by JETDS and Unit-Number .

SYSTEM_UNIT has a mandatory many-to-one "has a" relationship with SYSTEM and an optional many-to-one "has a" relationship with UNIT.

9. CABLE

The entity CABLE describes generic cables that connect units of a Tomahawk weapons system. It is identified by Cable-Number and includes Cable-Type, Cable-Length, Active-Wires, Connectors, Back-Shell-Kit, and Terminations.

CABLE has an optional one-to-many "has a" relationship with UNIT_CABLE.

10. UNIT_CABLE

The entity UNIT_CABLE describes a specific cable of a specific unit in a Tomahawk Weapon System. It is identified by Unit-Number and Cable-Number, both foreign keys.

UNIT_CABLE has optional many-to-one "has a" relationships with UNIT and CABLE.

11. PROBLEM

The entity PROBLEM describes a problem with a specific Tomahawk Weapon System, either discovered through a test or reported from the fleet. It is identified by Problem-Number and includes JETDS, Hull-Number, Test-Name, Test-Date, Problem-Description, Test-Procedure, Test-Revision, Test-Part-Step, Action-Taken, Date-Action-Taken, Date-Discovered, Problem-Coordinator, Prepared-By, and Prepared-By-Date.

PROBLEM has an optional one-to-many "has a" relationship with ACTION_ITEM, supertype/subtype "is a" relationships with FLEET_PROBLEM, TEST_PROBLEM, INCO_ANOMALY, and EXTERNAL_TEST_PROBLEM, an optional one-to-one "prepares a" relationship with PERSON, an optional many-to-one "coordinates a" relationship with PERSON, and an optional many-to-one "has a" relationship with SHIP_TEST.

12. FLEET_PROBLEM

The entity FLEET_PROBLEM describes a problem with a Tomahawk Weapon System discovered in the fleet. It is identified by FPR-Number and includes Related-

Problem-Number, Time-Meter, Site-Manager-Signature, Date-Site-Manager, Problem-Impact, Agency-Proposing-Solution, TCG-Software-Version, LCG-Software-Version, TEPEE-Software-Version, VLS-Software-Version, JOTS-Software-Version, STR-Number, Assigned-To, Date-Assigned, Closed-By, and Date-Closed.

FLEET_PROBLEM has a subtype/supertype "is a" relationship with PROBLEM, an optional many-to-one "proposes a" relationship with ORGANIZATION, and optional "signs a," "closes a," and "assigned to" relationships with PERSON.

13. TEST_PROBLEM

The entity TEST_PROBLEM describes a problem with a Tomahawk Weapon System discovered through a Tomahawk community test. It is identified by Problem-Number and includes TPR-File-Number, TPR-Serial-Number, Individual-Contacted, Date-Contacted, Date-Needed, Problem-Type, Problem-Impact, Solution, Solution-Approval, Disposition, Passed-To-Shipyard, Approved-TD, and Date-TD.

TEST_PROBLEM has a subtype/supertype "is a" relationship with PROBLEM, and optional many-to-one "approves a (TD)" and "contacted about" relationships with PERSON.

14. INCO_ANOMALY

The entity INCO_ANOMALY describes a problem with a Tomahawk weapons system discovered through an Installation and Check Out Test. It is identified by Problem-Number and includes Category, Hazard, Failure-Detected, Failure-Isolated, Fault-Code, Time-Meter, PCB-Signature, Date-PCB, Site-Manager-Approval, Date-Site-Manager, Old-Part-Number, Old-Part-Serial-Number, New-Part-Number, New-Part-Serial-Number, NHA-Part-Number, NHA-Part-Serial-Number, Related-IAR-Number, Agency-Providing-Solution, Approved-TDA, Date-TDA, Approved-TD, and Date-TD.

INCO_ANOMALY has a subtype/supertype "is a" relationship with PROBLEM, an optional many-to-one "provides a" relationship with ORGANIZATION, and optional many-to-one "approves a (TDA)," "approves a (TD)," "approves a (Site Manager)," and "signs a" relationships with PERSON.

15. EXTERNAL_PROBLEM

The entity EXTERNAL_TEST_PROBLEM describes a problem with a Tomahawk weapons system discovered through a test conducted by an agency external to the Tomahawk engineering support community. It is identified by Problem-Number and includes Local-Report-Number, Organizational-Element, Recommendations, Priority, Category, Configured-Item, Related-Test-Report, Urgency, Corrected-Version, Status, Status-Date.

EXTERNAL_PROBLEM has a subtype/supertype "is a" relationship with PROBLEM and an optional one-to-many "relates to a" relationship with PROBLEM.

16. ACTION_ITEM

The entity ACTION_ITEM describes an assignment to a person to perform a task within the Tomahawk engineering support community. It is identified by Action-Item-Number and includes Problem-Number, Source, Monitor, Action-Description, Requestor, Assigned-To, Assigned-To-Date, Due-Date, Status, Close-Date, Assignee, Remarks, and Project.

ACTION_ITEM has optional many-to-one "assigns a," "assigned a," "requests a," and "monitors a" relationships with PERSON and an optional many-to-one "has a" relationship with PROBLEM.

17. ORGANIZATION

The entity ORGANIZATION describes a command within the Tomahawk engineering support community. It is identified by UIC and includes Command-Name, and Acronym.

ORGANIZATION has a mandatory one-to-many "has a" relationship with PERSON, an optional one-to-many "proposes a" relationship with FLEET_PROBLEM, and an optional one-to-many "provides a" relationship with INCO_ANOMALY.

18. PERSON

The entity PERSON describes a member of a Tomahawk engineering support organization. It is identified by ID-Number and includes Organization, First-Name, Last-Name, Rank, Code, Classified-Address, Unclassified-Address, Phone-Number, Secure-Phone-Number, FAX-Number, FAX-Check-Number, Department, Building, Level, Room, Mail-Code, Job-Title, Remarks, and Responsibilities.

PERSON has optional one-to-many "coordinates a" and "prepares a" relationships with PROBLEM; optional one-to-many "assigns a," "assigned a," "requests a," and "monitors a" relationships with ACTION_ITEM; optional one-to-many "approves a (TD)" and "contacted about" relationships with TEST_PROBLEM; optional one-to-many "approves a (TDA)," "approves a (TD)," "approves a (site manager)," and "signs a" relationships with INCO_ANOMALY; an optional one-to-many "originates a" relationship with NETWORK_TROUBLE; optional one-to-many "chairs a" and "poc of a" relationships with EVENT; a mandatory many-to-one "has a" relationship with ORGANIZATION; and optional one-to-many "closes a," "assigned to," and "signs a" relationships with FLEET_PROBLEM.

19. EVENT

The entity EVENT describes a scheduled meeting or event within the Tomahawk engineering support community. It is identified by Chairperson, Start-Date, and Start-Time and includes Point-Of-Contact, End-Date, Mini-Subject, Location, Security-Classification, Clearance-Address, Remarks, Agenda.

EVENT has optional many-to-one "chairs a" and "poc of a" relationships with PERSON.

20. NETWORK_TROUBLE

The entity NETWORK_TROUBLE describes a problem with the TEXN network experienced and submitted by persons within the Tomahawk engineering support community. It is identified by NTR-Number and includes Originator, Brief-Description, Version, Computer, Suggested-Priority, Related-NTR, Description, NTR-Status, Trouble-

Code, Assigned-Priority, Fixed-Version, Closure-Date, and System-Administrator-Remarks.

NETWORK_TROUBLE has an optional many-to-one "originates a" relationship with PERSON.

D. OVERLAP WITHIN TEXN

Overlap within a database can be characterized as data or functional. Data overlap is information that is redundantly represented in more than one table of a database. Functional overlap occurs when different applications of a database system redundantly perform the same function. TEXN contained both types of overlap.

1. Data Overlap

Since the database structure in TEXN is a flat file design, there were many data redundancies and overlaps. A prime example of this overlap was information on persons which existed in every database table except Cable Run Sheets (CAB). For example, a preparer was included in the databases of Action Items, Test Problem Reports, and External Certification Test Reporting. Another example of data overlap included data on ships, which existed in Action Items, Test Problem Reports, Fleet Problem Reports, External Certification Test Reporting, and Cable Run Sheet databases. When generating the entity-relationship diagram for TEXN, these data redundancies were eliminated.

2. Functional Overlap

We discovered one functional overlap within TEXN. There is the possibility of three different applications, Test Problem Reporting, Fleet Problem Reporting, and External Test Problem Reporting, reporting the same problem. The difference between the three reports is the reporting source. However, this functional overlap is a function of the Tomahawk engineering support community's reporting system, not a TEXN anomaly.

In the next chapter, we will examine the TIMES conceptual data model.

VI. TIMES CONCEPTUAL DATA MODEL

This chapter describes the conceptual data model for TIMES. It is organized as follows. Section A provides the motivation for developing a conceptual model for TIMES. Section B describes the process of reverse engineering the TIMES entity-relationship model. Section C examines TIMES entities, defining them and listing their key attributes and relationships. Finally, section D briefly examines the data and functional overlaps within TIMES.

A. BACKGROUND

TIMES was developed using a formal design methodology. The design documentation that was available included an E-R diagram generated by Oracle CASE and a complete data dictionary, broken down by each table in the database. We wanted to be sure that we modeled TIMES in its current state. To facilitate the comparative analysis of the current state of TIMES with the TEXN conceptual model, the reverse data engineering utility of the CASE tool used for this project, System Architect 3.0, was used to create an E-R data model from the current TIMES schema definitions.

B. DEVELOPING THE TIMES ENTITY-RELATIONSHIP MODEL

To implement the reverse engineering of TIMES, we needed a current Data Definition Language (DDL) file. Obtaining this file proved slow and difficult. Some of these problems were due to lack of technical knowledge, while others were political in nature.

1. Reverse Engineering Process

Several parts of the DDL needed some manipulation before the tool could read it properly. For example, information about table space allocation, as well as table view and sequence definition were removed because they are not needed to generate the conceptual data model. There were also some system specific commands that did not relate to the data definitions, and were therefore removed.

Once the syntax problems were resolved, we discovered that the DDL did not include any information about the primary or foreign keys for the TIMES tables. This resulted in a schema with no keys and no relationships.

The primary and foreign keys for all the tables were added to the DDL using the TIMES data dictionary and additional information from NSWC DD on the structure of new tables. (NSWC DD, Jul 1993) Among the new tables were several simple look-up tables that were not relevant to the data model. These tables were eliminated from the DDL to simplify processing and enhance our understanding of the model.

2. Verifying Entities and Attributes

Once the new E-R model was developed, the next step was to verify the entities and their attributes against the documentation. This was a fairly simple process in which we compared the current model against the original one and noted any differences. We found some changes in the implementation from the documented model, including the addition of new entities.

There are three instances where a supertype-subtype relationship has been eliminated. In two of these instances, the data from the supertype was incorporated into each of its subtypes, and in the remaining instance data from the subtypes was incorporated into the supertype. These instances are detailed below.

The entity ACTION_ITEM contained two subtypes in the original design: SOFTWARE_TROUBLE_ACTION_ITEM and MISSION_PLANNING_RFA. Each of these two entities now includes the attributes originally included in ACTION_ITEM, as well as their own attributes. The same change was made to the CHANGE_REQUEST entity. SOFTWARE_TROUBLE and ANOMALY were subtypes, but each now contains the attributes of the supertype CHANGE_REQUEST. The entity SOFTWARE_VERSION was a supertype, but it now contains the attributes of its subtypes, BUILD and BASELINE.

There are several new entities that are not included in the original documentation. They are:

ANOMALY_ASSESSMENT
ANOMALY_MAIL
ANOMALY_PREFIX
CM_MAIL
DOCUMENT_CHANGE_PACKAGE
SQA_ACTIVITY
SQA_RECOMMENDATION
SQA_RESPONSE
STR_PREFIX

Descriptions of each of these entities, as well as their primary and foreign keys, were obtained from NSWC DD.

DOCUMENT_CHANGE_PACKAGE is not an entirely new entity. It is defined in the data dictionary, but is not present in the original E-R diagram or design report. (NSWC DD, Apr 1993) This entity would be more accurately described as a view, combining the attributes of DOCUMENT and CHANGE_PACKAGE and relating them to SOFTWARE_TROUBLE. We were unable to determine why it was made into a physical table; there does not seem to be any performance enhancement to warrant this change.

The attributes for each entity were compared against the data dictionary. No significant differences were found. Identifying attributes was made difficult and confusing by the re-use of names from one entity to the next. Id-number is a good example of such confusion; it was listed as an attribute, usually the primary key, in several entities. But in each it represented a semantically different concept.

3. Verifying the Relationships

The relationships defined in the new E-R diagram were then verified. This was done by comparing them to the original data model, the data dictionary, and new information from NSWC DD. Only a few differences were identified.

The relationships involving DOCUMENT_CHANGE_PACKAGE were significantly changed. There are two one-to-many relationships with SOFTWARE_TROUBLE. Each of these relationships displaces a respective relationship between DOCUMENT, CHANGE_PACKAGE and SOFTWARE_TROUBLE.

The new model showed no relationships to the entity STATUS_HISTORY, while the original data model clearly shows a relationship between SOFTWARE_TROUBLE_STATUS and STATUS_HISTORY. The lack of a relationship was due to the data dictionary, which showed no foreign keys present in STATUS_HISTORY. We determined this to be a flaw in the data dictionary and added the foreign keys, resulting in the appropriate relationship.

4. Generating the Entity-Relationship Diagram

System Architect generated an initial E-R diagram. The entities, attributes, and relationships were verified, refined, and changed as necessary. There were some aspects of the software generated E-R diagram that needed to be changed.

Several entities were duplicated in the resulting diagram. Each of these duplicates was deleted and the relationships drawn to the correct entity. Two entities, TEST and DOCUMENT_CHANGE_PACKAGE, did not have any attributes. These attributes were entered manually.

The layout out of the software generated model did not have the entities located in the most logical and readable manner. The layout was modified for readability and to closely resemble the original E-R diagram. This change made comparisons with the original model easier, and reduced the clutter of the 136 relationship connection lines.

C. ENTITY DESCRIPTIONS

In describing the reverse engineered entity-relationship model of TIMES, the main entities and their relationships are discussed briefly.

1. ANOMALY

An ANOMALY is an error or defect against software that is not under official Configuration Management (CM) control. It is identified by id-number.

ANOMALY has an optional one-to-many "generated from" relationship with SOFTWARE_TROUBLE, a mandatory one-to-many "associated with" relationship with ANOMALY_ASSESSMENT, an optional many-to-one "written against" relationship with HARDWARE_SUITE, a mandatory many-to-one "originated by" relationship with ORGANIZATION, a mandatory many-to-one "written by" relationship with SITE, a mandatory many-to-one "containing" relationship with SPECIFIC_ACTIVITY, a mandatory many-to-one "applicable to" relationship with SYSTEM, and an optional many-to-one "referenced in" relationship with TEST.

2. ANOMALY_ASSESSMENT

An ANOMALY_ASSESSMENT is an assessment of an Anomaly used to determine that a requested change is necessary. The Anomaly Assessment documents operator impacts, recommended priority, recommended course of action, and work estimates. It is identified by id-number, anom-id-number, and anomaly-group, and includes assessment-date, created-by-name, creation-date, assessor-name, assessor-phone-number, comments-text, last-changed-date, last-changed-by-name, change-difficulty-code, error-category-code, est-affected-easy-nbr, operational-factor-cd, est-affected-hard-nbr, operator-impact-text, recommendation-text, work-est-hours, est-affected-med-nbr, prblty-of-occrnce-code, est-sloc, swvr-name-introduced-in, swvr-sys-code, swvr-prgm-code, swvr-swvr-name, swvr-swvr-sys-code, swvr-swvr-prgm-code, and anom-asm-id.

ANOMALY_ASSESSMENT has a mandatory many-to-one "associated with" relationship with ANOMALY and an optional many-to-one "introduced in" relationship with SOFTWARE_VERSION.

3. ANOMALY_MAIL

ANOMALY_MAIL is a mail message about an anomaly. It is identified by id-number and includes release-date, type, prgm-code, anomaly-group.

ANOMALY_MAIL has mandatory many-to-one "references" relationship with PROGRAM and a mandatory many-to-one "references" relationship with ANOMALY_PREFIX.

4. ANOMALY_PREFIX

An ANOMALY_PREFIX describes the different categories to which an ANOMALY can belong to. It is identified by anomaly-group, prgm-code, and prefix and includes retired and sequence-no.

ANOMALY_PREFIX has a mandatory one-to-many "included in" relationship with ANOMALY, a mandatory many-to-one "refers to" relationship with PROGRAM, and a mandatory one-to-many "referenced by" relationship with ANOMALY_MAIL.

5. ASSESSMENT

An ASSESSMENT is an evaluation of a SOFTWARE_TROUBLE used to determine that a requested change is necessary. It is identified by id-number and str-id-number.

ASSESSMENT has a mandatory many-to-one "associated with" relationship with SOFTWARE_TROUBLE and an optional many-to-one "introduced in" relationship with SOFTWARE_VERSION.

6. CHANGE_PACKAGE

A CHANGE_PACKAGE is a set of change pages that are grouped together and are incorporated into a document that is under CM control. It is identified by id-number, doc-id-number, and doc-revision-number.

CHANGE_PACKAGE has an optional one-to-many "traceable to" relationship with SOFTWARE_TROUBLE_STATUS, a mandatory many-to-one "incorporated in"

relationship with DOCUMENT, and an optional one-to-many "referenced in" relationship with SQA_RECOMMENDATION.

7. CM_MAIL

CM_MAIL describes a specific announcement of a new document change package. It is identified by id-number and includes doc-id-number, doc-revision-number, and release-date.

CM_MAIL has a mandatory many-to-one "announces" relationship with DOCUMENT.

8. DOCUMENT

A DOCUMENT is approved documentation describing a SYSTEM's or item's characteristics. It is identified by id-number and revision-number.

DOCUMENT has a mandatory one-to-many "changed by" relationship with CHANGE_PACKAGE, a mandatory many-to-one "pertaining to" relationship with SYSTEM, and an optional one-to-many "referenced in" relationship with SQA_RECOMMENDATION.

9. DOCUMENT_CHANGE_PACKAGE

A DOCUMENT_CHANGE_PACKAGE describes all DOCUMENTS and CHANGE_PACKAGES maintained in TIMES and is used for validation of the DOCUMENTS affected on a SOFTWARE_TROUBLE. It is identified by doc-id-number, doc-revision-number, and cpkg-id-number.

DOCUMENT_CHANGE_PACKAGE has an optional one-to-many "referenced by" relationship with SOFTWARE_TROUBLE.

10. FUNCTIONAL_AREA

A FUNCTIONAL_AREA is a distinct group of SYSTEM performance requirements which may be grouped together as a function. It is identified by code.

FUNCTIONAL_AREA has a mandatory one-to-many "basis for" relationship with FUNCTIONAL_AREA_AFFECTED.

11. FUNCTIONAL_AREA_AFFECTED

FUNCTIONAL_AREA_AFFECTED is the unique value, word, or set of characters assigned that identifies the FUNCTIONAL_AREA where a SOFTWARE_TROUBLE or ANOMALY has been detected. It is identified by functional-area-code, str-anml-id-number, and str-anml-type.

FUNCTIONAL_AREA_AFFECTED has a mandatory many-to-one "based on" relationship with FUNCTIONAL_AREA, a mandatory many-to-one "on" relationship with SOFTWARE_TROUBLE, and a mandatory many-to-one "on" relationship with ANOMALY.

12. GENERAL_ACTIVITY

GENERAL_ACTIVITY is the general value that represents the activity being done when a SOFTWARE_TROUBLE report or ANOMALY was written. It is identified by name.

GENERAL_ACTIVITY has a mandatory one-to-many "made up of" relationship with SPECIFIC_ACTIVITY.

13. GROUP

GROUP identifies a collection of people by the type of function they perform in regard to their assigned Action Item. It is identified by name.

GROUP has an optional one-to-many "assigned" relationship with SOFTWARE_TROUBLE_ACTION_ITEM.

14. HARDWARE_SUITE

The entity HARDWARE_SUITE is each hardware suite or system where a SOFTWARE_TROUBLE or ANOMALY has occurred. It is identified by name.

HARDWARE_SUITE has an optional one-to-many "referenced on" relationship with SOFTWARE_TROUBLE and an optional one-to-many "referenced on" relationship with ANOMALY.

15. MEETING

A MEETING is an organization of technical and administrative personnel that come together for a specific purpose. They could be reviewing and evaluating proposed changes to baselines, initiating and tracking configuration changes, recommending the product's readiness for release and other issues. It is identified by id-number, mtgt-code, and prgm-code.

MEETING has a mandatory one-to-many "the source of" relationship with MISSION_PLANNING_RFA, an optional one-to-many "the modifier of" relationship with SOFTWARE_TROUBLE_STATUS, a mandatory many-to-one "verified" relationship with MEETING_TYPE, a mandatory many-to-one "referencing" relationship with PROGRAM, and a mandatory one-to-many "represented by" relationship with MEETING_ROLE.

16. MEETING_ROLE

A MEETING_ROLE is the function of the PERSON in the MEETING. It is identified by mtg-id-number, mtgt-code, mtg-prgm-code, and prsn-seq-number.

MEETING_ROLE has a mandatory many-to-one "represented in" relationship with MEETING and a mandatory many-to-one "performed by" relationship with PERSON.

17. MEETING_TYPE

A MEETING_TYPE is the list of valid types of MEETINGS held. It is identified by code.

MEETING_TYPE has a mandatory one-to-many "used as" relationship with MEETING.

18. MISSION_PLANNING_RFA

A MISSION_PLANNING_RFA is an interface related action item. It is identified by seq-number.

MISSION_PLANNING_RFA has an optional many-to-one "assigned to" relationship with ORGANIZATION, a mandatory many-to-one "originated by" relationship with ORGANIZATION, a mandatory many-to-one "initiated by" relationship with MEETING, and an optional many-to-one "closed by" relationship with PERSON.

19. ORGANIZATION

An ORGANIZATION is a number of PERSONS or GROUPS having specific responsibilities and united for a particular purpose. It is identified by code.

ORGANIZATION has a mandatory one-to-many "the originator of" relationship with MISSION_PLANNING_RFA, an optional one-to-many "assigned" relationship with MISSION_PLANNING_RFA, a mandatory one-to-many "originating" relationship with REQUEST_FOR_ACTION, a mandatory one-to-many "the originator of" relationship with SOFTWARE_TROUBLE, a mandatory one-to-many "the originator of" relationship with ANOMALY, and a mandatory one-to-many "the employer of" relationship with PERSON.

20. PERSON

A PERSON is an individual that is identified in the system as an attendee or database user. Most persons tracked in TIMES will be users; however, many persons are relevant to TIMES that are not database users. It is identified by seq-number.

PERSON has a mandatory one-to-many "acting in" relationship with ROLE, an optional many-to-one "defaulted to" relationship with PRINTERS, a mandatory one-to-many "acting in" relationship with MEETING_ROLE, a mandatory many-to-one "employed by" relationship with ORGANIZATION, an optional one-to-many "closing" relationship with MISSION_PLANNING_RFA, and optional one-to-many "approving authority for" and "assigned to" relationships with REQUEST_FOR_ACTION.

21. PROGRAM

A PROGRAM is a system of systems with a defined purpose and scope. It is identified by code.

PROGRAM has a mandatory one-to-many "represented by" relationship with SYSTEM, a mandatory one-to-many "assigned" relationship with ROLE, a mandatory one-to-many "referenced by" relationship with MEETING, and an optional one-to-many "assigned" relationship with PERSON.

22. RELATED_SOFTWARE_TROUBLE

A RELATED_SOFTWARE_TROUBLE is a software trouble related to other software troubles. It is identified by str-id-number and rlst-id-number.

RELATED_SOFTWARE_TROUBLE has mandatory many-to-one "under" and "on" relationships with SOFTWARE_TROUBLE.

23. REQUEST_FOR_ACTION

A REQUEST_FOR_ACTION is an assigned action resulting from a review or audit, such as corrections to design or requirement specifications and directions for special technical reviews. It is identified by control-number.

REQUEST_FOR_ACTION has a mandatory many-to-one "against" relationship with DOCUMENT, a mandatory one-to-many "the source of" relationship with RESULTING_SOFTWARE_TROUBLE, a mandatory many-to-one "originated by" relationship with ORGANIZATION, and optional many-to-one "approved by" and "assignment of" relationships with PERSON.

24. RESULTING_SOFTWARE_TROUBLE

The entity RESULTING_SOFTWARE_TROUBLE is a cross reference between REQUEST_FOR_ACTION and SOFTWARE_TROUBLE. It is identified by str-id-number and rfa-control-number.

RESULTING_SOFTWARE_TROUBLE has a mandatory many-to-one "based on" relationship with REQUEST_FOR_ACTION and a mandatory many-to-one "based on" relationship with SOFTWARE_TROUBLE.

25. ROLE

A ROLE is the function of the PERSON in the GROUP. It is identified by prsn-seq-nbr, prgm-code, and rtyp-code.

ROLE has a mandatory many-to-one "performed by" relationship with PERSON, a mandatory many-to-one "assigned to" relationship with PROGRAM, a mandatory many-to-one "of" relationship with ROLE_TYPE, and an optional many-to-one "representing" relationship with SYSTEM.

26. ROLE_TYPE

A ROLE_TYPE is the function of a ROLE in explaining the reason for an action taken. It is identified by code.

ROLE_TYPE has a mandatory one-to-many "the classification of" relationship with ROLE.

27. SITE

A SITE is a ship or land-based location which contains or is in support of a Tomahawk Cruise Missile System of a particular baseline. It is identified by code.

SITE has a mandatory one-to-many "referenced on" relationship with SOFTWARE_TROUBLE and a mandatory one-to-many "referenced on" relationship with ANOMALY.

28. SOFTWARE_TROUBLE

A SOFTWARE_TROUBLE is a problem recorded against software under official CM control. It is identified by id-number.

SOFTWARE_TROUBLE has a mandatory one-to-many "the basis for" relationship with RESULTING_SOFTWARE_TROUBLE, a mandatory one-to-many "subject of" relationship with SOFTWARE_TROUBLE_ACTION_ITEM, mandatory one-to-many "referenced by" and "related to" relationships with RELATED_SOFTWARE_TROUBLE, a mandatory one-to-many "affected by" relationship with FUNCTIONAL_AREA_AFFECTED, a mandatory many-to-one "originated by" relationship with ORGANIZATION, a mandatory many-to-one "containing" relationship with SPECIFIC_ACTIVITY, an optional one-to-one "generated from" relationship with ANOMALY, a mandatory many-to-one "written by" relationship with SITE, a mandatory many-to-one "applicable to" relationship with SYSTEM, an optional recursive one-to-many "duplicated by" relationship with SOFTWARE_TROUBLE, a mandatory one-to-many "source of" relationship with ASSESSMENT, a mandatory many-to-one "against" relationship with SOFTWARE_VERSION, an optional many-to-one "based on" relationship with DOCUMENT_CHANGE_PACKAGE, an optional many-to-one "has a" relationship with STR_PREFIX, and a mandatory one-to-many "reason for" relationship with SOFTWARE_TROUBLE_STATUS.

29. SOFTWARE_TROUBLE_ACTION_ITEM

A SOFTWARE_TROUBLE_ACTION_ITEM is a SOFTWARE_TROUBLE task originating from a Software Control Review Board (SCRB) or Configuration Control Board (CCB) which is assigned for resolution. It is identified by str-id-number and seq-number.

SOFTWARE_TROUBLE_ACTION_ITEM has a mandatory many-to-one "referencing" relationship with SOFTWARE_TROUBLE and a mandatory many-to-one "assigned to" relationship with GROUP.

30. SOFTWARE_TROUBLE_STATUS

The entity SOFTWARE_TROUBLE_STATUS describes the current status of a SOFTWARE_TROUBLE. It is identified by swts-str-id-number, swts-swvr-name, swts-

swvr-prgm-code, swts-swvr-sys-code, swts-swvr-swvr-name, swts-swvr-swvr-prgm-code, and swts-swvr-swvr-sys-code.

SOFTWARE_TROUBLE_STATUS has a mandatory on-to-many "contained in" relationship with STATUS_HISTORY, an optional many-to-one "modified by" relationship with MEETING, a mandatory many-to-one "for" relationship with SOFTWARE_TROUBLE, an optional many-to-one "fixed by" relationship with CHANGE_PACKAGE, an optional many-to-one "fixed by" relationship with SOFTWARE_VERSION, a mandatory many-to-one "statused against" relationship with SOFTWARE_VERSION, and a mandatory many-to-one "verified as being" relationship with STATUS.

31. SOFTWARE_VERSION

A SOFTWARE_VERSION is any defined and documented Configuration Management set of software that may be statused on a trouble report. It is identified by name, prgm-code, sys-code, swvr-name, swvr-prgm-code, and swvr-sys-code.

SOFTWARE_VERSION has a mandatory one-to-many "on" relationship with SOFTWARE_TROUBLE, optional one-to-many "introducing" and "the source of" relationships with ASSESSMENT, optional one-to-many "introducing" and "the source of" relationships with ANOMALY_ASSESSMENT, a mandatory recursive many-to-one "part of" relationship with SOFTWARE_VERSION, a mandatory many-to-one "contained in" relationship with SYSTEM, a mandatory one-to-many "reference on" relationship with SOFTWARE_TROUBLE_STATUS, an optional one-to-many "resolved by" relationship with SOFTWARE_TROUBLE_STATUS, and a mandatory many-to-one "referenced on" relationship with SQA_RECOMMENDATION.

32. SQA_ACTIVITY

The entity SQA_ACTIVITY is a group responsible for conducting software quality assurance. It is identified by activity, prgm-code, and sys-code, and includes prefix, retired, and sequence-no.

SQA_ACTIVITY has a mandatory one-to-many "contained in" relationship with SQA_RECOMMENDATION and a mandatory many-to-one "references" relationship with SYSTEM.

33. SQA_RECOMMENDATION

The entity SQA_RECOMMENDATIONS describes the software quality assurance recommendations for a SOFTWARE_VERSION. It is identified by id-number and includes prgm-code, sys-code, created-by-name, creation-date, rec-prefix, rec-number, originator-name, origination-date, swvr-swvr-name, swvr-name, title-text, doc-id-number, doc-revision-number, cpkg-id-number, sqa-activities, satv-name, urgency-code, resolution-text, review-date, act-assigned, response-date, status, update-date, rec-type, category, cost, benefit, submit-flag, and rec-num.

SQA_RECOMMENDATION has a mandatory many-to-one "references" relationship with DOCUMENT, a mandatory many-to-one "references" relationship with SYSTEM, a mandatory many-to-one "includes" relationship with SQA_RESPONSE, a mandatory many-to-one "includes" relationship with SPECIFIC_ACTIVITY, and a mandatory many-to-one "results in" relationship with SQA_RESPONSE.

34. SQA_RESPONSE

The entity SQA_RESPONSE describes the response to a proposed SQA_RECOMMENDATION. It is identified by id-number and includes rec-id-number, created-by-name, response-date, responders-name, response, comments, status, submit-flag, and resp-id.

SQA_RESPONSE has a mandatory many-to-one "responds to" relationship with SQA_RECOMMENDATION.

35. SPECIFIC_ACTIVITY

A SPECIFIC_ACTIVITY is the activity being done when a SOFTWARE_TROUBLE report is written. It is identified by name and gatv-name.

SPECIFIC_ACTIVITY has a mandatory one-to-many "contained on" relationship with SOFTWARE_TROUBLE, a mandatory one-to-many "contained on" relationship with ANOMALY, a mandatory one-to-many "contained on" relationship with SQA_RECOMMENDATION, and a mandatory many-to-one "part of" relationship with GENERAL_ACTIVITY.

36. STATUS

A STATUS is the list of valid statuses assigned to a SOFTWARE_TROUBLE. It is identified by code.

STATUS has a mandatory one-to-many "used as" relationship with SOFTWARE_TROUBLE_STATUS.

37. STATUS_HISTORY

A STATUS_HISTORY is the progression of values that track the history of the status changes associated with a SOFTWARE_TROUBLE. It is identified by str-id-number, status-code, change-date, swvr-name, swvr-prgm-code, swvr-sys-code, swvr-swvr-name, swvr-swvr-prgm-code, and swvr-swvr-sys-code.

STATUS_HISTORY has a mandatory one-to-many "associated with" relationship with SOFTWARE_TROUBLE_STATUS.

38. STR_PREFIX

A STR_PREFIX defines the category to which a SOFTWARE_TROUBLE belongs. It is identified by prefix and prgm-code and includes retired and sequence-no.

STR_PREFIX has a mandatory many-to-one "refers to" relationship with PROGRAM and a mandatory one-to-many "contained in" relationship with SOFTWARE_TROUBLE.

39. SYSTEM

The entity SYSTEM includes all equipment, related facilities, material, software, services, and personnel required for its operation and support to the degree that it can be

considered a self-sufficient item in its intended operational environment. It is identified by code and prgm-code.

SYSTEM has a mandatory many-to-one "represented in" relationship with PROGRAM, an optional one-to-many "represented by" relationship with ROLE, a mandatory one-to-many "composed of" relationship with SOFTWARE_VERSION, a mandatory one-to-many "the subject of" relationship with DOCUMENT, a mandatory one-to-many "referenced on" relationship with SOFTWARE_TROUBLE, a mandatory one-to-many "referenced on" relationship with ANOMALY, and a mandatory one-to-many "referenced in" relationship with SQA_ACTIVITY.

40. TEST

A TEST is used to evaluate and determine the cause of the problem. It is identified by code.

TEST has a mandatory one-to-many "referenced on" relationship with SOFTWARE_TROUBLE and a mandatory one-to-many "referenced on" relationship with ANOMALY.

D. OVERLAP WITHIN TIMES

Overlap within a database can be characterized as data or functional. Data overlap is information that is redundantly represented in more than one table. Functional overlap occurs when applications within a database redundantly perform the same function. TIMES was found to contain an instance of data overlap.

The creation of the DOCUMENT_CHANGE_PACKAGE entity results in data overlap. Every attribute of this entity is the same as one in either DOCUMENT or CHANGE_PACKAGE. There are no relationships between DOCUMENT_CHANGE_PACKAGE and the other two entities, making it possible for update anomalies to occur. This overlap was left in the E-R model, but will be removed prior to implementing any changes.

VII. TEXN AND TIMES DATA OVERLAP

In this chapter, we examine the data overlap between TEXN and TIMES databases. We use the framework presented in Chapter II to identify and resolve the schema conflicts of the two databases.

A. OVERLAPPING ENTITIES

Overlapping entities are entities that represent identical real-world objects and have the same name. The two main overlapping entities in both TEXN and TIMES are PERSON and TEST.

TEXN entity PERSON is an individual that is identified in the TEXN database. Examples include test directors, report preparers, problem coordinators, assigners of actions, assigned action recipients, points of contact requestors, and monitors. TIMES entity PERSON is an individual that is identified in the system as an attendee or database user. The attendees include points of contact, program managers, and Change Review Board members.

TEXN entity TEST is defined as a type of test performed on a Tomahawk Weapons System. TIMES entity TEST is defined as a type of test used to evaluate and determine the cause of a problem. Both definitions are identical indicating a data overlap between the two databases.

B. ENTITY LEVEL CONFLICTS

Schema level conflicts develop from the conceptual structure and definition of the databases. These conflicts are divided into entity level conflicts, attribute level conflicts, and entity attribute conflicts. Conflicts between equivalent entities of heterogeneous databases are called entity level conflicts. Entity level conflicts include naming, entity structure, and constraint conflicts.

1. Naming Conflicts

Entity level conflicts exist when like entities in heterogeneous databases have differing names (synonyms), or differing entities have the same name (homonyms).

Entity synonyms and homonyms for TEXN and TIMES databases follow (see Tables 7-1 and 7-2).

Database	Entity Name	Entity Definition
TEXN	SHIP	A ship or site that contains a Tomahawk weapon system.
TIMES	SITE	A ship or land-based location which contains or is in support of a Tomahawk Cruise Missile System.
TEXN	EVENT	A meeting, training session, or conference held within the Tomahawk support community.
TIMES	MEETING	An organization of technical and administrative personnel that come together for a specific purpose.
TEXN	ACTION_ITEM	A task assigned to an individual or group of individuals within the Tomahawk support community.
TIMES	REQUEST FOR ACTION	An assigned action resulting from a review or audit.

Table 7-1. TEXN/TIMES Entity Level Synonyms

Database	Entity	Entity Definition
TEXN	ORGANIZATION	A command within the Tomahawk support community.
TIMES	ORGANIZATION	A group that has specific responsibilities and are united for a specific purpose.

Table 7-2. TEXN/TIMES Entity Level Homonyms.

Entity level synonyms are similar entities that have different names. TEXN entity SHIP is a synonym with TIMES entity SITE. TEXN entity SHIP is defined as a location or platform that contains and supports a Tomahawk Weapons System (TWS). TIMES entity SITE is defined similarly as a ship or land-based location which contains or is in support of a Tomahawk Cruise Missile System of a particular baseline.

TEXN entity EVENT is a synonym with TIMES entity MEETING. TEXN entity EVENT is defined as a meeting, training session, or conference held within the Tomahawk support community. TIMES entity MEETING is defined as an organization of technical and administrative personnel that come together for a specific purpose and hence is similar to TEXN's EVENT entity. These groups can review and evaluate proposed changes to baselines, initiate and track configuration changes, recommend the product's readiness for release, and other issues.

TEXN entity ACTION_ITEM is a synonym with TIMES entity REQUEST FOR ACTION. TEXN entity ACTION_ITEM is defined as a task assigned to an individual or group of individuals within the Tomahawk support community. TIMES entity REQUEST FOR ACTION is defined as an assigned action resulting from a review or audit, such as corrections to a design or requirement specifications and directions for a special technical review. Both definitions are similar albeit the different names of their respective entities.

In contrast, entity level homonyms are different entities that have the same name. TEXN and TIMES entities ORGANIZATION are homonyms. While TEXN entity ORGANIZATION is defined as a command within the Tomahawk support community the TIMES entity ORGANIZATION is defined as a group that has specific responsibilities and are united for a specific purpose.

2. Structure Conflicts

Entity structure conflicts can be caused by incomplete attribute sets for like entities. This occurs when one database does not include attributes in an equivalent entity in another database because the attributes were not considered of interest. This conflict occurs in the synonym entity comparisons between TEXN and TIMES that follow.

TIMES entity PERSON has attributes that TEXN entity PERSON does not. These attributes include Middle Initial Identifier, Suffix Name, Phone Cntry Code Number, Autovon Number, Email Identifier, Oracle User Number, and Default Program Code.

Similarly the attributes Chair-Person, Point-Of-Contact, End-Date, Mini-Subject, Subject, Security-Classification, Clearance-Address, and Remarks are included in TEXN entity EVENT but not the TIMES entity MEETING.

TEXN entity SHIP_SYSTEM has the attribute Serial-Number that TIMES entity HARDWARE SUITE does not.

C. ATTRIBUTE LEVEL CONFLICTS

Attribute level conflicts describe inconsistencies between equivalent attributes and include name, structure, and constraint conflicts.

1. Naming Conflicts

Similar to entity naming conflicts, attribute level conflicts occur when like attributes in heterogeneous databases have differing names (synonyms), or differing attributes have the same name (homonyms). The following are attribute synonyms for TEXN and TIMES databases (see Table 7-3):

- Attribute Hull-Number of TEXN entity SHIP and attribute Code of TIMES entity SITE. Hull-Number is defined as the ship's hull number or site's code. Code identifies the standard abbreviation of a site or ship, for example CG 47, and is therefore synonymous to Code.

- Attribute Ship-Name of TEXN entity SHIP and attribute Name of TIMES entity SITE. Ship-Name is defined as the name of the ship or site. Name similarly identifies the full name of an afloat or land based location, for example USS TICONDEROGA.

- Attribute Start-Date of TEXN entity EVENT and attribute Date of TIMES entity MEETING. Both Start-Date and Date identify the date an event or meeting is to take place.

Attribute	Entity	Database	Attribute Definition
Hull-Number	SHIP	TEXN	A ship's hull number or site's code.
Code	SITE	TIMES	A site's code.
Ship-Name	SHIP	TEXN	A name of the ship or site.
Name	SITE	TIMES	A name of a site.
Start-Date	EVENT	TEXN	A date an event is to take place.
Date	MEETING	TIMES	A date a meeting is to take place.
Start-Time	EVENT	TEXN	A time that an event is to take place.
Time	MEETING	TIMES	A time that a meeting is to take place.
Location	EVENT	TEXN	A location of an event.
Location Name	MEETING	TIMES	A location of a meeting.
ID-Number	PERSON	TEXN	A number that identifies a person.
Sequence Number	PERSON	TIMES	A number that identifies a person.
Rank	PERSON	TEXN	Identifies a person's rank.
Military-Rank	PERSON	TIMES	Identifies a person's military rank.
Phone-Number	PERSON	TEXN	A person's telephone number.
Telephone Number	PERSON	TIMES	A person's telephone number.
JETDS	SYSTEM	TEXN	Nomenclature of a TWS.
Code	SYSTEM	TIMES	A code that represents a TWS.
Test-Name	TEST	TEXN	A name of a test performed on a TWS.
Code	TEST	TIMES	A code of a test performed on a TWS.
Test-Description	TEST	TEXN	A description of a test.
Description Test	TEST	TIMES	A description of a test

Table 7-3. TEXN/TIMES Attribute Level Synonyms.

- Attribute Start-Time of TEXN entity EVENT and attribute Time of TIMES entity MEETING. Both Start-Time and Time identify the time that the meeting or event is to take place.

- Attribute Location of TEXN entity EVENT and attribute Location Name of TIMES entity MEETING. Both Location and Location Name identify the location of the meeting or event.

- Attribute ID-Number of TEXN entity PERSON and attribute Sequence Number of TIMES entity PERSON. Both ID-Number and Sequence Number are unique numbers that identify a person. In TIMES, the Sequence Numbers are automatically generated by the system.

- Attribute Rank of TEXN entity PERSON and attribute Military Rank of TIMES entity PERSON. Both Rank and Military Rank identifies the person's military rank.

- Attribute Phone-Number of TEXN entity PERSON and attribute Telephone Number of TIMES entity PERSON. Both Phone-Number and Telephone Number identifies the telephone number at which the person may be reached.

- Attribute JETDS of TEXN entity SYSTEM and attribute Code of TIMES entity SYSTEM. Both JETDS and Code uniquely identify a type of TWS.

- Attribute Test-Name of TEXN entity TEST and attribute Code of TIMES entity TEST. Both Test-Name and Code uniquely identify a type of test performed on a TWS when a problem was discovered.

- Attribute Test-Description of TEXN entity TEST and attribute Description Text of TIMES entity TEST. Both Test-Description and Description Text are narrative text that describe a type of test.

2. Constraint Conflicts

Attribute conflicts occur because the detailed description of synonym attributes differs in type, length, or range. When a classification of two attributes are different, e.g., character versus numeric, it is called a type clash. When the character field lengths are different, it is labeled a length clash. When the allowable set of values for the same attribute are different, it is called a range clash. (Kamel, 1994) TEXN and TIMES attribute conflicts follow (see Table 7-4).

Type clashes occur between TIMES and TEXN attribute level synonyms. Attribute Start-Time of TEXN entity EVENT is of type numeric while attribute Time of TIMES entity MEETING is of type character. Attribute ID-Number of TEXN entity PERSON is of type character while attribute Sequence Number of TIMES entity PERSON is of type numeric. Attribute Phone-Number of TEXN entity PERSON is of type numeric while attribute Telephone Number of TIMES entity PERSON is of type character.

D. ENTITY ATTRIBUTE LEVEL CONFLICTS

When equivalent information is represented as an entity in one database and an attribute in another, it is called an entity attribute conflict. This occurs because of conceptual design decisions made by different design teams. TEXN and TIMES entity attribute conflicts follow.

TIMES entity SOFTWARE VERSION represents the same information as the attributes, TCG-Software-Version, LCG-Software-Version, TEPEE-Software-Version, VLS-Software-Version, and JOTS-Software-Version of TEXN entity FLEET_PROBLEM. SOFTWARE VERSION identifies a defined and documented configuration management set of software on a trouble report. TCG-Software-Version identifies the software version of Track Control Group (TCG) in use when the problem occurred. This also applies to the software versions of Launch Control Group (LCG), Tomahawk Engagement Planning Evaluation Enhancement (TEPEE), Vertical Launch System (VLS), and Joint Operational Tactical System (JOTS).

Attribute	Entity	Database	Data Type	Data Length
Hull-Number	SHIP	TEXN	Character	6
Code	SITE	TIMES	Character	25
Ship-Name	SHIP	TEXN	Character	22
Name	SITE	TIMES	Character	80
Start-Date	EVENT	TEXN	Character	8
Date	MEETING	TIMES	Character	7
Start-Time	EVENT	TEXN	Numeric	4
Time	MEETING	TIMES	Character	5
Location	EVENT	TEXN	Character	16
Location Name	MEETING	TIMES	Character	30
ID-Number	PERSON	TEXN	Character	10
Sequence Number	PERSON	TIMES	Numeric	8
Rank	PERSON	TEXN	Character	16
Military Rank	PERSON	TIMES	Character	6
Phone-Number	PERSON	TEXN	Numeric	10
Telephone Number	PERSON	TIMES	Character	13
JETDS	SYSTEM	TEXN	Character	14
Code	SYSTEM	TIMES	Character	20
Test-Name	TEST	TEXN	Character	22
Code	TEST	TIMES	Character	20
Test-Description	TEST	TEXN	Character	78
Description Text	TEST	TIMES	Character	240

Table 7-4. TEXN/TIMES Attribute Type/Length Clash Conflicts.

TIMES entity ASSESSMENT represents the same information as the attributes Problem-Description of TEXN entity PROBLEM, Problem-Impact of TEXN entities TEST_PROBLEM and FLEET_PROBLEM, and Solution of TEXN entity TEST_PROBLEM. ASSESSMENT is an evaluation of a software trouble used to determine that a requested change is necessary and includes description of the problem, operator impacts, and recommended courses of action. Problem-Description is a description of a problem experienced with a TWS, which can include software troubles. Problem-Impact describes the impact of the problem. Problem-Solution describes recommended courses of action.

In the next chapter we will examine the alternate solutions for the integration of the Tomahawk support community's heterogeneous databases.

VIII. DATABASE INTEGRATION ALTERNATIVES

This chapter describes the alternative approaches for the integration of the Tomahawk support community's heterogeneous databases. It is organized as follows. Section A examines the full merging approach and discusses its advantages and disadvantages. Sections B and C explore the advantages and disadvantages of the tightly coupled federated database approach and the loosely coupled federated database approach, respectively. Finally, section D examines the data warehouse approach and discusses its pros and cons.

A. FULLY MERGED DATABASE

1. Description

In this method, database integration is performed by combining relations from disparate databases into a single physically unified database. This is accomplished through the integration of the data dictionaries, database management systems, and the data itself. The end result is a fully unified, centrally located database.

The integration of data dictionaries of heterogeneous databases requires a semantically rich integrating model, such as the entity-relation diagram, to describe, identify, and resolve conflicts of the different component database systems. (Kamel 1994) Schemas are generated for each heterogeneous database and then compared against each other in a top down fashion to identify redundancies and conflicts. Schema and data level conflicts are categorized and resolved. The schema and associated data dictionary are then fully integrated, using a single database management system to form a fully unified database.

2. Advantages

The full merging approach allows a complete and clean design of a database that combines data from two or more databases. This approach produces a new, fully functional database system that is free of conflicts using a desirable database management system.

3. Disadvantages

The full merging approach has some disadvantages. First, it does not preserve the autonomy of the separate heterogeneous databases. This loss of autonomy dilutes the support that the database provides for the organization for which it was initially designed. Organizations are reluctant to relinquish control of components for which they have been responsible.

Second, the full merging approach is often not economically or politically feasible. The cost of developing and implementing a completely new database system can be prohibitive and may outweigh the benefits provided by the new, fully merged database. Furthermore, organizations are inherently resistant to change, and may attempt to block such a structural change.

B. FEDERATED DATABASE: TIGHTLY COUPLED APPROACH

1. Description

The second option for the integration of the Tomahawk support community's heterogeneous databases is a federated database system (FDBS) using a tightly coupled approach. Sometimes referred to as a multidatabase approach, a federated database allows data integration without losing the autonomy of individual databases.

In the tightly coupled approach, each independent database becomes part of the federation. A global schema is created that represents the integration of the component local schemas. This global schema describes the data that can be accessed by the FDBS. A user interacts with the FDBS as if it is a single database.

A query is issued on the global schema and the data is retrieved from the different databases without the users direct knowledge. A global controller coordinates among the database components. It receives the query, breaks it down, and translates it into subqueries on the individual local schemas. The results of the subqueries are collected, data conflicts resolved, and the information properly formatted. The information is then sent back to the user.

The data dictionary of a federated database system is different from that of a single homogeneous database. A data dictionary usually describes the logical data structure as well as the underlying database. The FDBS data dictionary will present a global picture of the data, but will not describe the individual database structure of each local database. (Kamel & Ceruti, 1994)

2. Advantages

The primary advantage of the tightly coupled federated database system is that the component databases maintain their autonomy. This is a highly desirable situation as it provides both economic and political advantages.

Economically, both time and money can be saved by maintaining the autonomy of the component databases. Minimal hardware and software changes will be required at the local level. In today's environment of significant monetary constraints, implementation of a FDBS may be the most effective way to increase functionality without significantly increasing costs.

Additionally, if a current database system is productive, it is always politically difficult to implement radical changes. The users and administrators of the system will often provide significant resistance to changes to their system. Implementing a FDBS can alleviate much of the resistance, and still improve the interoperability and integration among the different database systems.

3. Disadvantages

To make the tightly coupled approach work, a global controller is required. It acts as the coordinator and translator among the component databases. This is a very complex component to design, implement, and maintain. The design and implementation of the controller can be as challenging as the design of a database management system.

The global schema can also be difficult to maintain. It is very sensitive to changes in the local schema. Without strong configuration control over the component databases, the global schema can quickly become inaccurate.

An organization with centralized control is sometimes required to make the tightly coupled approach work. This can infringe on the autonomy of the component databases, weakening one of the primary advantages of the tightly coupled approach.

C. FEDERATED DATABASE: LOOSELY COUPLED APPROACH

1. Description

In the loosely coupled approach, a global schema is not developed, requiring the users to be aware of the individual databases. The databases are accessed using a common language or a gateway. The language allows joining of data in different databases, broadcasting of queries over the different databases, exchange of data between databases, and the transformation of attribute values, units of measure, etc. (Kamel & Ceruti, 1994)

Along with the language, a complex web of application programming interfaces (APIs), drivers, stacks, and protocols must be defined and implemented to access the data. The complexity of this web is related to the level of heterogeneity among the databases. If all the systems are from a single commercial vendor, there are fewer problems than if the databases come from many different vendors. (Orfali, et al. 1994) In this environment, the user can access each of the different databases and the full functionality of the database will remain intact.

A gateway allows a database to be accessed remotely by a different system. It provides a pre-determined translation from some other environment to the local database environment. Using a gateway, commands given in an Oracle database can be translated for use by a DB2 database as illustrated in Figure 8-1. A gateway provided with the Oracle DBMS will translate commands and queries to a DB2 format. These commands are then sent to the DB2 server and processed. The results are returned to the Oracle DBMS and presented to the user. This process will require that the user have some knowledge of the structure of the DB2 database so the queries can be written correctly. Some DBMS features may not be available through a gateway. Typically, only an intersection set of the two DBMS features is available. (Orfali, et al. 1994)

Efforts are underway to implement standardized methods of cross database access. This would allow databases that are compliant with a given standard to communicate without specifically designed gateways. Once these standards emerge from the marketplace, the design and implementation of loosely coupled federated database systems will be much simpler and the functionality across different DBMSs will be enhanced.

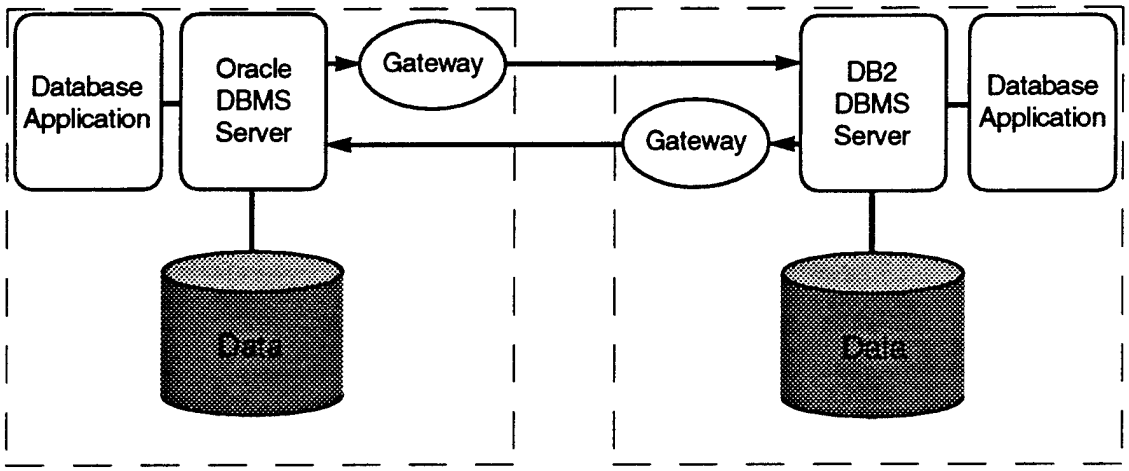


Figure 8-1. Loosely Coupled Federated Database.

2. Advantages

The loosely coupled FDBS is simpler to implement than the tightly coupled approach. There is no need for a global schema and the associated complexity. The autonomy of the component databases is maintained, giving the same economic and political advantages as the tightly coupled approach.

If a common DBMS is used for the federated database, users will have access to the different databases with the full functionality of each system intact.

3. Disadvantages

The lack of a global schema means that users must be knowledgeable about the structure and content of the other databases in the FDBS. This problem increases as the number of component databases increases.

Users may also be required to learn a new language. If gateways are not implemented to allow use of the local database interface, the user will need to learn a separate language for accessing the other component databases.

A gateway must exist for each of the different types of databases that will make up the FDBS. These gateways are provided by the vendor of each DBMS and are not an open architecture solution. Each vendor does not necessarily have a gateway for every other vendor and may not support full connectivity with each gateway. The gateway may only support data extracts and queries, not transaction processing. (Orfali, et al. 1994)

D. DATA WAREHOUSE

1. Description

Data warehouses organize, store, and summarize data required for analysis and decision support. The data warehouse provides a separate database for the integration of the data from separate operational database systems.

Data entering the warehouse comes from the operational environment, which can include multiple heterogeneous databases. Data warehouses aggregate data at multiple levels to support management's decision making processes.

A data warehouse is subject oriented, focusing on key enterprise areas of an organization and only includes data that will be used in decision support systems (DSS) processing. This focus on subject areas influences the design of the data and its structuring.

A data warehouse integrates data by selecting a standard format for data and then translating all data that enters the warehouse to this format. The data is integrated in many ways, including consistent naming conventions, encoding structures, and measurements of variables. The identical labeling of equivalent entities and attributes is an example of a consistent naming convention. Encoding structures represent attribute instances with a code, for instance the representation of gender being represented as a "Male" or "Female," "M" or "F," or "0" or "1." A consistent encoding structure would standardize on a single code, such as "M" and "F" for gender and translate all other codes to this structure when

integrating heterogeneous data. The translation of all attribute measurements, e.g., inches, centimeters, and feet, to a common measurement unit, e.g., centimeters, is an example of consistent measurement variables.

A data warehouse is time variant and is only accurate to the moment in time when the data was transferred to the warehouse. This differs from the operational database that is accurate at the moment of access. The information in the warehouse is a series of snapshots in time of the operational database, which provide a history of the organization's data, making it available for aggregation and trend analysis. Every key structure implicitly or explicitly contains an element of time to distinguish the data time variance. Once the data is extracted from the operational databases and stored in the data warehouse, it cannot be updated.

Since the information that is stored in the data warehouse cannot be changed, just added to, it is nonvolatile. There are only two types of operations performed on a data warehouse: the initial loading of the data and the access of the data. Data in a data warehouse is not updated, which eliminates the presence of data anomalies found in an operational database. Normalization of data is not required, therefore data can be structured to optimize the access of data.

2. Structure

Components of a data warehouse include meta data, current detail data, older detail data, lightly summarized data, and highly summarized data. The meta data contains the structure of the data, algorithms used for summarizing the data, and the mapping of the operational environment to the data warehouse. Current detail data is the most recently loaded data and is stored at the lowest level of granularity. Older detail data is stored at the equivalent level of data as the current detail data. Lightly and heavily summarized data are data distilled from the detailed data to different levels of compactness for trend analysis and decision support. An example of the contents of a data warehouse for a sales organization follows. Current level detailed data would consist of sales data from 1992, and old detailed sales data from 1982 to 1991. Lightly summarized data would represent regional sales

data by week from 1983 to 1993 and highly summarized data national sales by month from 1985 to 1993 (see Figure 8-2). (Inmon, 1995)

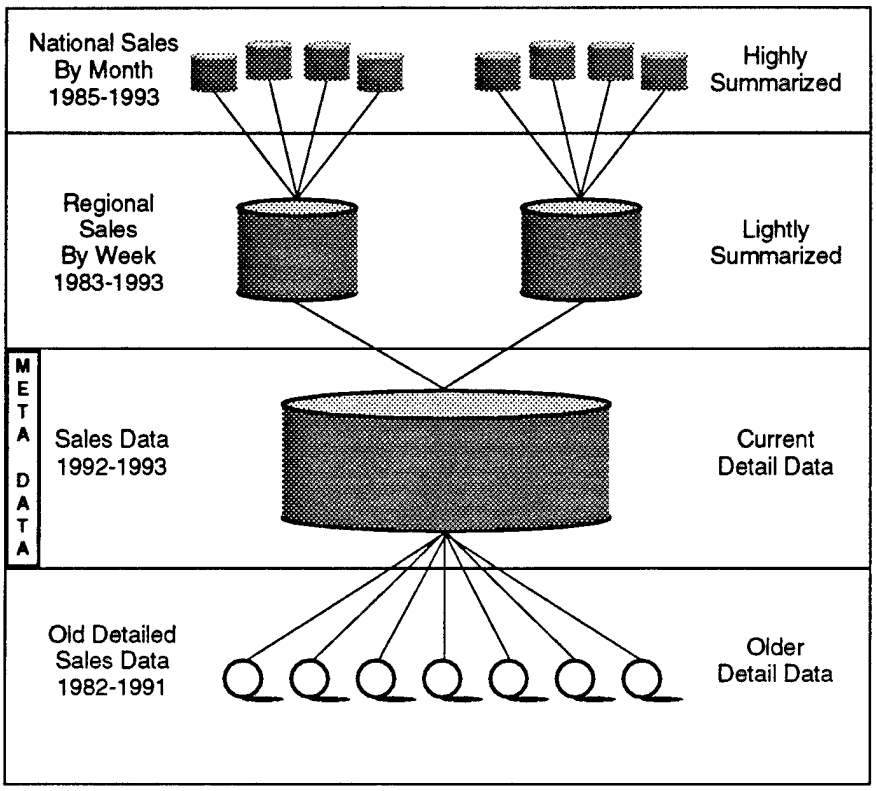


Figure 8-2. Data Warehouse After (Inmon, 1995).

Data flows into a data warehouse from the operational environment which can include many heterogeneous databases. This data resides in the current level detail until it is purged, is summarized, or archived.

3. Advantages

The data warehouse alternative for integrating heterogeneous databases provides many benefits, including evolutionary development, conflict resolution, enhanced system performance, and improved resource usage.

A data warehouse is developed through an evolutionary, step by step process which simplifies data integration.

Conflict resolution is a benefit provided by using a data warehouse to integrate heterogeneous databases. Since data from all operational databases in converted to a standardized format, conflicts between databases are resolved.

Data warehouses improve enterprise resource utilization. Database accesses for historic data are extremely burdensome on an operational transaction processing system. Transaction processing speed is critical for online transaction processing systems, while database access for historic data for analysis is less so. A large volume of database accesses for historic information can significantly reduce the processing speed of a transaction processing system. A data warehouse separates the transaction processing from database querying, improving system performance.

Data warehouses promote improved resource utilization. Online transaction processing systems store all data on relatively expensive disk storage to support the fast access required. The separate data elements of a data warehouse receive different levels of usage, and can therefore be deployed on different resources in order to optimize access speed versus the cost of the storage medium.

4. Disadvantages

The major disadvantage of a data warehouse is stale data. Since data in the data warehouse is not updated, the current detailed data may not represent the currency of the operational databases. The data in a data warehouse is only accurate to a point in time.

In the next chapter we will recommend an alternative for the integration of the Tomahawk community's heterogeneous databases.

IX. SUMMARY, RECOMMENDATIONS, AND LESSONS LEARNED

This chapter summarizes the thesis and discusses the results of the research. It is organized as follows. Section A summarizes the thesis and relates it to its initial objectives. Section B presents our recommendations for an integration strategy for the Tomahawk engineering community. Section C discusses the lessons learned while conducting this research. Section D outlines opportunities for future work.

A. SUMMARY

The primary goal of this research was to develop a methodology for identifying an integrating strategy and architecture for multiple heterogeneous databases within an organization. Initially, we expected to use a top down process for the development of that methodology. An initial methodology was developed for data gathering, analysis, and integration strategies and then applied to TEXN and TIMES. We discovered that this methodology was not comprehensive enough to support the goals of the thesis.

The development of the methodology became an iterative process that required revising and expanding steps to address previously overlooked or insufficiently covered aspects of the analysis. After each revision, the steps were re-applied to the analysis of TEXN and TIMES. The result was a six step process for determining the best approach for integrating data from diverse, distributed, heterogeneous databases. The process includes general guidelines for identifying the most appropriate integration solution based on degree of overlap.

The analysis of TEXN revealed a flat file structure. Re-engineering was necessary to develop a relational data model. This model allowed us to compare TEXN against TIMES to identify overlaps. The analysis of TIMES revealed a well designed relational database. Reverse engineering ensured the development of a conceptual data model that reflected the latest implementation and upgrade changes. The Entity-Relationship data modeling technique provided a common representation for comparison of the databases.

The comparison of the databases identified overlapping objects and semantic conflicts. The extent of overlap was found to be less than ten percent, indicating a minimal level of overlap.

Several approaches for integration of heterogeneous databases were evaluated, and their advantages and disadvantages identified. These approaches included data warehousing, tightly and loosely coupled federated databases, and a fully merged database. The evaluation allowed us to recommend the solution that was the most feasible and best meets the needs of the Tomahawk engineering community.

B. RECOMMENDATIONS

We recommend implementing a loosely coupled federated database integration strategy for the Tomahawk engineering community (see Figure 9-1).

1. Reasoning

The detailed study of TEXN and TIMES database structure revealed minimal overlap. A cursory examination of other Tomahawk community databases indicated minimal overlap between them as well. The loosely coupled federated approach is generally the best integration strategy for databases with minimal overlap.

This approach allows the individual databases to retain their autonomy. Minimal hardware and software changes are required while maintenance and administration remain local. Users gain greater access to data without having to learn a new system. In addition, this approach promotes stability and reduces political resistance.

The loosely coupled approach allows the integration of additional databases without changes to the schema. If a database management system (DBMS) standard has been chosen, a database using that DBMS can easily be incorporated into the federation. In a non-standardized environment, a database can be incorporated if a gateway exists for that DBMS.

Applications can be developed that will use the gateway to access the various databases in the federation. A user can implement a query based on knowledge of the data

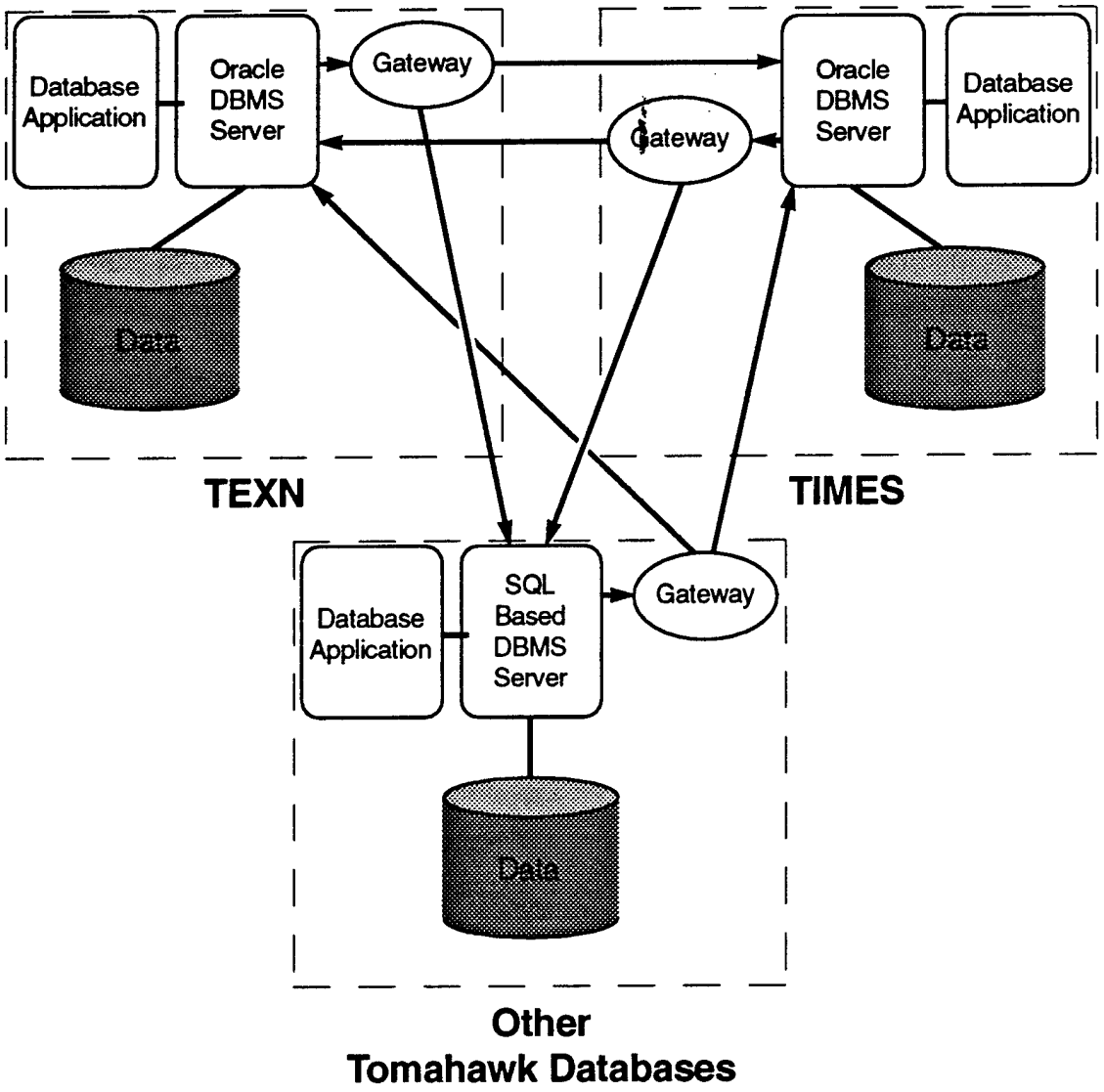


Figure 9-1. Proposed Database Integration Solution.

and schemas of the other databases. The gateway will handle the query by identifying the relevant databases and properly formatting the request. The query may apply to several different databases at once, allowing the user to access and use data in a manner that is not possible when using separate, standalone databases.

2. Implementation Details

To implement a loosely coupled federated database integration, the following actions are recommended.

To simplify the federated database, we recommend selecting Oracle as the standard DBMS for all future database implementations within the Tomahawk engineering community. If this is not feasible, only SQL based DBMSs for which a gateway exists should be used.

TEXN should be redesigned as a relational database: federated databases are greatly simplified by using a relational data structure. Data must be decoupled from applications and the conceptual data models re-engineered. This thesis has already produced a re-engineered conceptual data model for TEXN.

We recommend using an Oracle DBMS for the implementation of the re-engineered TEXN database and leaving it under the control of WCS-SEIA. Since TIMES uses Oracle, this would simplify the implementation of a loosely coupled federated database. FOXPRO could continue to be used in the relational implementation using Microsoft's ODBC to provide a gateway interface. This approach will still require a redesign of TEXN and increase the complexity of the federated database.

C. LESSONS LEARNED

Fostering cooperation is essential in a project with diverse stakeholders. These stakeholders must be included in the planning process and clearly understand the goals of the project. Concerns about the effects of the project on their area of responsibility must be allayed. We were unable to fully accomplish this due to the time and scope limits of our thesis.

The people responsible for TEXN and TIMES knew very little about the goals of our project. Everyone was cooperative but also apprehensive about how the results of our research would impact their areas of responsibility. The resulting confusion and resistance may have contributed to the delay we encountered in receiving data critical to the reverse engineering process for TIMES. Additional communication effort in the early stages of our research may have eliminated this problem.

We encountered aspects of the TEXN and TIMES systems in which the processes used did not seem logical or efficient. The scope of our research was limited to the integration of these two databases. It was evident that many applications within TEXN and TIMES were just automated manual processes. A business process review (BPR) would have been beneficial in improving process efficiency and effectiveness. Any project of the magnitude of Engineering 2000 should include a BPR in the early stages. Our data models were limited by the processes using the databases.

Some problems were encountered when working with the System Architect CASE tool. The reverse engineering utility used a data definition listing (DDL) as input to produce the E-R diagram. The DDL we received for TIMES could not be read by System Architect without modification. This involved removing extraneous information from the file. The documentation gave no guidance on the content of the DDL and a call to the technical assistance line was equally unproductive.

The DDL was also missing information about the primary and foreign keys for the database tables. It was unclear if this was a problem caused during generation of the DDL or a problem with Oracle's implementation of DDLs. Research on all aspects of the use and generation of DDLs for reverse engineering should be conducted early in the project.

There were several aspects of the TEXN and TIMES databases that made the analysis more difficult and could cause problems for the database integration efforts. Common data fields, such as name and phone number, had inconsistent structures within each of the databases. The name fields were not always the same length, and not all phone numbers included an area code. Standard data definitions should be implemented for data

fields common to both databases. These definitions should be based on current Department of Defense (DoD) efforts for data element standardization.

In TIMES, the data fields "id_number," "code," and others were used several times with different definitions, depending on the table in which they were contained. Data inconsistencies of this type should be eliminated and more descriptive data names implemented before federating the databases.

The "date" data fields used by TEXN and TIMES will cause a problem regardless of the decision on database integration. All date fields use a two digit year. When the year 2000 arrives, queries based on year will be unreliable. All dates should be changed to a four digit year.

The analysis process was further complicated by the lack of documentation of changes made to the databases. The conceptual data model for TIMES would have been outdated if we had not used reverse engineering to ensure we modeled the current status of the database.

D. FUTURE WORK

While this thesis develops a solid methodology for the integration of heterogeneous databases, it has only been applied to two databases within the Tomahawk engineering community. Further work should be conducted to apply the proposed methodology to the other databases within the community. The methodology should also be applied outside the Engineering 2000 project to verify its applicability and improve its content.

The database integration process should be completed. Functional requirements must be identified and hardware, interface, and other standards defined. A migration plan should then be implemented that will enable development and implementation of the integrated database.

It may be possible to develop a methodology to aid in the completion of each of these steps. Research should be conducted to develop the methodology. The final result would be a structured process to take an organization completely through a database integration process from evaluation to full implementation.

APPENDIX A. DATABASE QUESTIONNAIRE

Site visits were conducted at the operating locations of both TEXN and TIMES. A questionnaire was developed to guide the interviews of the administrators and users of the databases. The questions are listed below.

Development History

- What system was used before TEXN (TIMES) was developed?
  - Was it a single system or multiple systems?
  - Was it a manual or an automated system?
  - Was it a database system or some other type of automated system?
- Why was TEXN (TIMES) developed?
  - Was it in response to a specific request or part of a larger automation process?
- When was TEXN (TIMES) developed?
- Have there been upgrades or changes to the database system?
  - Are these changes documented?
  - Is more than one version in use?
- Are there any long term changes planned for TEXN (TIMES)?
  - What are they?
- Are there any long term plans for hardware changes or upgrades?
  - What are they?
  - Are these changes designed for TEXN (TIMES) or general in nature?

- There is a version of TEXN for both Macintosh and DOS platforms.
  - Why are there two versions?
  - Is the Mac version ported from DOS or was a separate version written?

System Requirements

- What is the hardware required for each database?
- What is the software required for each database?
- Are hardware limitations restricting the use of the databases?
- Is there a current need for remote access to the database?
- Is there a future need for remote access?
- What are the time requirements for data updates (do they require online processing, or is batch processing sufficient)?

Volumes and Frequencies

- Is system usage monitored by the operating system?
  - Can we have access to the usage reports?
- How often is each(the) database accessed:
  - For an update?
  - For a query?
- How often is each table accessed:
  - For an update?
  - For a query?

- How often is each report generated?
    - What tables are accessed for each report?
- How many records are currently contained in each database?
- How much disk space does each database use?
- What are the projections for future size of the database?
    - How were these determined?
- What are the performance requirements for the database?
    - access speed:
        - update speed:
        - query speed:
    - How were these determined?
    - Are they being met?

Data

- Is there a conceptual model of the database (e.g., E-R diagram, IDEF1X)?
    - Is this model available only in hard copy or in digital format?
    - What modeling tool was used to produce the model?
    - Can we get a copy of the model?
- What is the underlying model of the database (relational, hierarchical, network)

- What is the content and structure of the meta-data (data dictionary)?
  - Data name definitions.
  - Data size definitions.
  - Data type definitions (e.g. alphanumeric, numeric).
  - Unique identifier fields.
  - Other domain definitions (e.g. min/max values, required fields).
  - Where is this structure documented?
- How were the data definitions determined?
  - Are they consistent with common usage throughout the organization?
    - Throughout the Navy?
  - Are they consistent within TEXN (TIMES)?
  - Are they consistent between TEXN and TIMES?
- How many tables are there for each database?
- How many attributes are there in each table?
- Is the database normalized?
  - What normal form is it in?
  - Is the database restricted to this level by the design structure or for performance reasons?

- There are several databases used within TEXN.
    - Is the same data present in different databases (data overlap)?
    - Is it entered separately for each database?
    - How are the Mac and DOS versions of databases kept consistent?
- Is there data overlap between TEXN and TIMES?
- Are there any controlled redundancies within the database?
- How is data integrity maintained?
    - How is access to records controlled?
        - Can several users access at once?
    - How are updates to records controlled?
        - Can several users update at once?

Applications

- What components make up the database system?
    - Application programs.
- Define the components of database management system.
    - What applications manipulate the database (querying, updating, generating reports)?
    - Are these functions performed by separate software components or by a single, integrated database management system?
    - Are these programs general purpose commercial-off-the-shelf (COTS) software or special purpose software?

- What user interfaces does the database system provide? (query languages, menu-driven interface, programming languages, natural languages, etc.)
  - Query languages (casual users).
  - Menu-driven interface (naive users).
  - Programming language interface (application programmers).
- Does the database system provide multiple views for end users?
- What functions does the database provide?
- What applications or functions generate data that is stored in the database?
- What are the outputs generated from the database (reports, etc.)?
- What applications access the data in the database?
- What other tools or systems would like to use the database?
- Are multiple levels of access required to the database?
- How is the data stored (online, batch, archival,etc.)?
- Is there functional overlap within each database? Between databases?
- Does this language support SQL queries?

Administration & Maintenance

- Is there a database administrator?
  - Who is it?
- Does the database administrator also maintain the database?
  - If not, who maintains the database?

- Who maintains the applications that use the database?
  - Have these people received training in TEXN (TIMES)?
- What type of user documentation exists for the database?

Users

- Who are the primary users?
  - Individuals?
  - Organizations?
- How would you categorize the database end users?
  - Casual (occasionally access the database, but they need different information each time they access it).
  - Parametric (their job function revolves around querying and updating the database).
  - Sophisticated (thoroughly familiar with the DBMS facilities, use it to meet their complex requirements).
- What percentages of the whole are each group of end users?
- What organizations use the database?
- What organizations administer and control database content?

Security

- What security functions are implemented in the database?
- What levels of data are in the database (unclassified, confidential, sensitive,etc)?
- Does the database contain several levels of access to users?

- Is there a database security program?
- Who maintains the database security program?

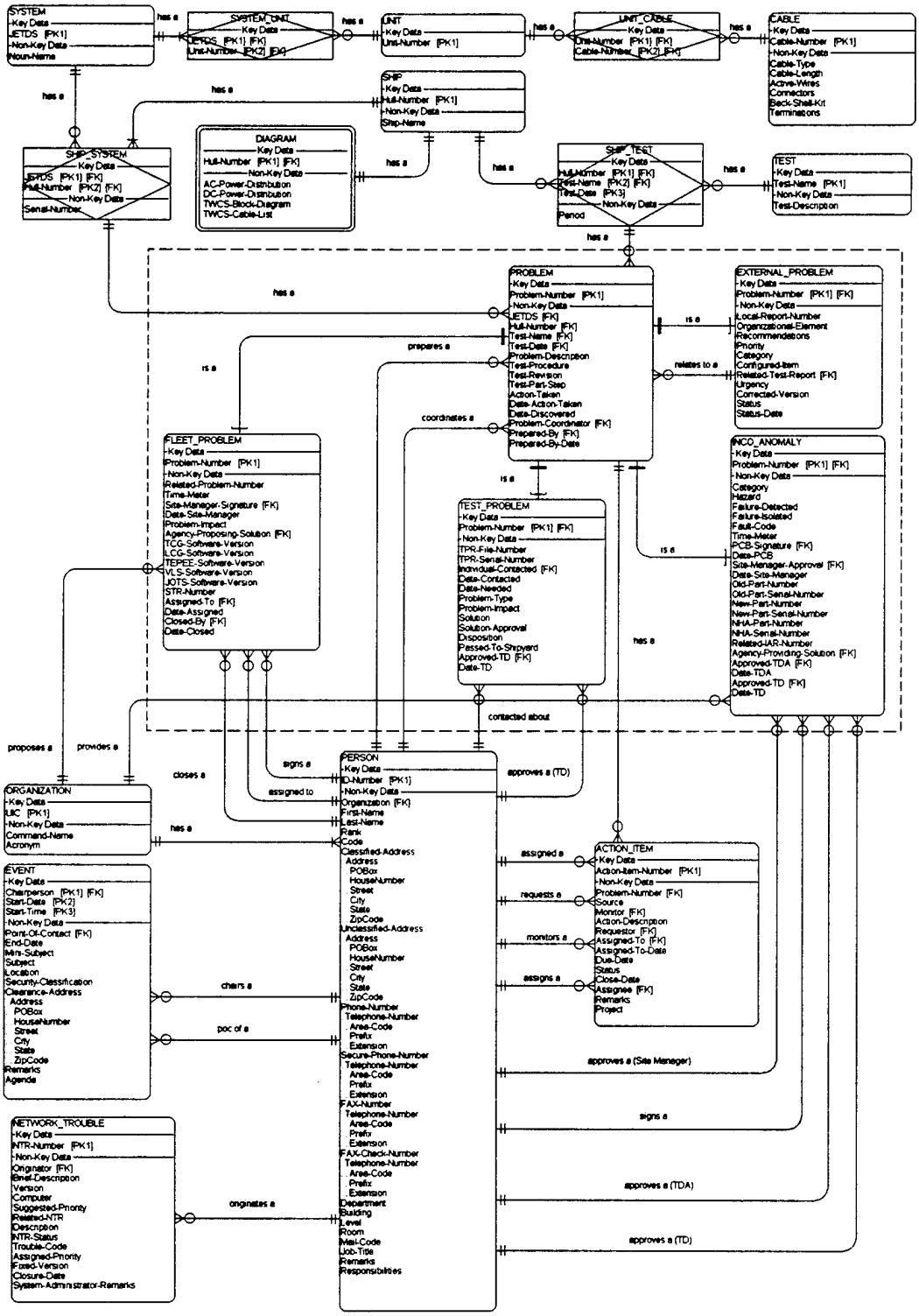
Backup and Recovery

- What backup procedures does the database system employ?
- What recovery procedures does the database use?
- How much data (time-wise) could be lost?
- What is the impact of the worst case scenario?

Concurrency Control

- What concurrency control functions are implemented in the database?

APPENDIX B. TEXN ENTITY-RELATIONSHIP DIAGRAM



APPENDIX C. TEXN DATA DEFINITIONS

Listed below are the data definitions for the re-engineered TEXN relational model. The definitions are grouped by entity. Each attribute is listed and the data type, length, and description are given.

ACTION_ITEM

PRIMARY KEYS: Action-Item-Number

FOREIGN KEYS:

Assigned-To	PERSON.ID-Number
Assignee	PERSON.ID-Number
Monitor	PERSON.ID-Number
Problem-Number	PROBLEM.Problem-Number
Requestor	PERSON.ID-Number

ATTRIBUTES:

Name	Type	Length
Action-Description	TEXT	78

Briefly describes the action that needs to be done.

Action-Item-Number	TEXT	12

Uniquely identifies the action assigned.

Assigned-To	TEXT	10

Identifies the ID number of the individual for whom the action item is assigned.

Assigned-To-Date	DATE	8

Identifies the date the action item was assigned.

Assignee	TEXT	10

Identifies the ID number of the individual who assigned the action item.

Close-Date	DATE	8

Identifies the date when the action item was closed.

Due-Date	DATE	8

Identifies the date when the action is to be completed.

Monitor	TEXT	10

Identifies the ID number of the individual who is looking after the task, but not necessarily responsible for the task.

Problem-Number	TEXT	9

Identifies the platform and problem sequence number assigned to the problem.

Project	TEXT	15

Describes the latest developments and the progress of the action item.

Remarks	TEXT	78

Describes the latest developments and the progress of the action item.

Requestor	TEXT	10

Identifies the ID number of the individual who requested the action.

Source	TEXT	27

Identifies where the action item was assigned.

Status	TEXT	8

Describes the status of the action item.

CABLE

PRIMARY KEYS: Cable-Number

FOREIGN KEYS: None

ATTRIBUTES:

Name	Type	Length
Active-Wires	NUMERIC	2

Identifies the number of active wires.

Back-Shell-Kit	TEXT	30

Identifies the part number of the backshell kit.

Cable-Length	NUMERIC	3

Identifies the length of the cable.

Cable-Number	TEXT	15

Identifies the cable designator number.

Cable-Type	TEXT	9

Identifies the part number of the cable.

Connectors	TEXT	20

Identifies the part number of the connector.

Terminations	TEXT	8

Identifies the termination file providing the pinout information of a particular cable.

DIAGRAM

PRIMARY KEYS: Hull-Number

FOREIGN KEYS:

Hull-Number SHIP.Hull-Number

ATTRIBUTES:

Name	Type	Length
AC-Power-Distribution	TEXT	10

Identifies the NAVSEA drawing number of the AC power distribution for a particular hull.

DC-Power-Distribution	TEXT	10

Identifies the NAVSEA drawing number of the DC power distribution for a particular hull.

Hull-Number	TEXT	6

Identifies the Platform.

TWCS-Block-Diagram	TEXT	10

Identifies the NAVSEA drawing number of the TWCS block diagram for a particular hull.

TWCS-Cable-List	TEXT	10

Identifies the NAVSEA drawing number of the TWCS cable list for a particular hull.

EVENT

PRIMARY KEYS: Chairperson

Start-Date

Start-Time

FOREIGN KEYS:

Chairperson	PERSON.ID-Number
Point-Of-Contact	PERSON.ID-Number

ATTRIBUTES:

Name	Type	Length
Agenda	TEXT	78

Lists up to one page of text on the main points of discussion or timeline of the meeting.

Chairperson	TEXT	10

Identifies the ID number of the person that will chair the meeting or has primary responsibilities for the event.

Clearance-Address	ADDRESS	67

Identifies the address to send clearance information required for attendance of the meeting or event.

End-Date	DATE	8

Identifies the scheduled event end date.

Location	TEXT	16

Identifies the short common name for the location of the meeting or event.

Mini-Subject	TEXT	80

Identifies the main point of discussion or the overall theme for the meeting or event in short form.

Point-Of-Contact	TEXT	10

Identifies the person designated as the point-of-contact (POC) for the event.

Remarks	TEXT	78

Contains short remarks on the meeting or event.

Security Classification	TEXT	2

Identifies the clearance required for attendance at the meeting or event.

Start-Date	DATE	8

Identifies the scheduled event begin date.

Start-Time	NUMERIC	4

Identifies the scheduled time for the event to begin in military time.

Subject	TEXT	80

Identifies the main point of discussion or overall theme for the meeting or event.

EXTERNAL_PROBLEM

PRIMARY KEYS: Problem-Number

FOREIGN KEYS:

Problem-Number PROBLEM.Problem-Number

Related-Test-Report PROBLEM.Problem-Number

ATTRIBUTES:

Name	Type	Length
Category	TEXT	1

Identifies the category of the problem, e.g., naval communication, tactical issue, TWCS hardware item, etc.

Configured-Item	TEXT	8

Identifies the problem authority of the configuration, e.g., TAC DOC, OP-3594, PQS, OPTASK F, etc.

Corrected-Version	TEXT	8

Identifies the software version that will contain the fix to the problem.

Local-Report-Number	TEXT	8

Identifies the test agency's local report number.

Organizational-Element	TEXT	3

Identifies the organizational subdivision within TWCS, e.g., TWCS B/L Software ENGINEERING, TWCS SURFACE SHIP REQUIREMENTS, etc.

Priority	NUMERIC	1

Indicates the priority associated with problem resolution: 1-Fix before fleet release, 2-Fix in follow on build, 3-Nice to have capability.

Problem-Number	TEXT	9

Identifies the platform and problem sequence number assigned to the problem.

Related-Test-Report	TEXT	9

Identifies any related problem test reports.

Status	TEXT	8

Identifies the status of the problem, e.g., OPEN, CLOSED, VALID, etc.

Status-Date	DATE	8

Identifies the date of the last update.

Urgency	TEXT	3

Describes the urgency of the problem resolution: L-Low, M-H - Medium to High, H-High.

FLEET_PROBLEM

PRIMARY KEYS: Problem-Number

FOREIGN KEYS:

Agency-Providing-Solution	ORGANIZATION.UIC
Assigned-To	PERSON.ID-Number
Closed-By	PERSON.ID-Number
Site-Manager-Signature	PERSON.ID-Number

ATTRIBUTES:

Name	Type	Length
Agency-Proposing-Solution	TEXT	24

Identifies the UIC of the agency that proposed the problem solution.

Assigned-To	TEXT	10

Identifies the ID number of the person assigned to the problem.

Closed-By	TEXT	10

Identifies the ID number of the person who closed the problem.

Date-Assigned	DATE	8

Identifies the date that the problem was assigned.

Date-Closed	DATE	8

Identifies the date the problem was closed.

Date-Site-Manager	DATE	8

Identifies the date that the Site Manager signed the FPR.

JOTS-Software-Version	TEXT	7

Identifies the JOTS software version of the involved in the problem.

LCG-Software-Version	TEXT	7

Identifies the LCG version of the software involved in the problem.

Problem-Impact	TEXT	78

Describes the impact on other hulls of the FPR.

Problem-Number	TEXT	10

Uniquely identifies the fleet problem, by hull number and sequential serial number.

Related-Problem-Number	TEXT	10

Identifies any related problem numbers.

Site-Manager-Signature	TEXT	10

Identifies the ID Number of the Site Manager that approved the FPR.

STR-Number	TEXT	23

Identifies any associated STR with the problem.

TCG-Software-Version	TEXT	7

Identifies the TCG version of the software involved in the problem.

TEPEE-Software-Version	TEXT	7

Identifies the TEPEE version of the software involved in the problem.

Time-Meter	NUMERIC	14

Identifies the time meter reading of the JETDS item at the point of problem discovery.

VLS-Software-Version	TEXT	10

Identifies the VLS version of the software involved in the problem.

INCO_ANOMALY_PROBLEM

PRIMARY KEYS: Problem-Number

FOREIGN KEYS:

Agency-Providing-Solution	ORGANIZATION.UIC
Approved-TD	PERSON.ID-Number
Approved-TDA	PERSON.ID-Number
PCB-Signature	PERSON.ID-Number
Problem-Number	PROBLEM.Problem-Number
Site-Manager-Approval	PERSON.ID-Number

ATTRIBUTES:

Name	Type	Length
Agency-Providing-Solution	TEXT	24

Identifies the UIC of the agency that provided the problem solution.

Approved-TD	TEXT	10

Identifies the ID number of the Test Director that recommended closure for the IAR.

Approved-TDA	TEXT	10

Identifies the ID number of the TDA that concurred with the IAR.

Category	TEXT	1

Identifies one of six categories for the IAR/TPR: D-Design, H-Hardware, T-Test Procedure, S-Software, I-Installation, and O-Other.

Date-PCB	DATE	8

Identifies the date the Problem Control Board (PCB) reviewed the IAR.

Date-Site-Manager	DATE	8

Identifies the data that the site manager signed the IAR.

Date-TD	DATE	8

Identifies the date that the test director approved the final closure of the IAR.

Date-TDA	DATE	8

Identifies the date that the TDA concurred with the IAR.

Failure-Detected	TEXT	1

Identifies that the problem was detected by a BIT/BYTE test.

Failure-Isolated	TEXT	1

Identifies that the problem was isolated by a BIT/BYTE test.

Fault-Code	TEXT	14

Identifies the fault code displayed by the BIT/BYTE test.

Hazard	TEXT	3

Used when the IAR/TPR describes a problem or condition which has caused, or has the potential to cause injury to personnel and/or serious damage to material.

New-Part-Number	TEXT	25

Identifies the part number including dash numbers and revision levels for the replacement part.

New-Part-Serial-Number	TEXT	14

Identifies the serial number of the replacement part.

NHA-Part-Number TEXT 25

Identifies the part number of the next higher assembly (NHA).

NHA-Serial-Number TEXT 14

Identifies the serial number of the next higher assembly (NHA).

Old-Part-Number TEXT 25

Identifies the part number, including dash numbers and revision levels, of the part removed.

Old-Part-Serial-Number TEXT 14

Identifies the serial number of the part removed.

PCB-Signature TEXT 10

Identifies the ID number of the IAR Problem Control Board (PCB) chairman.

Problem-Number TEXT 9

Identifies the platform and sequence number assigned to the problem.

Related-IAR-Number TEXT 25

Identifies the IAR number of other problems affected by the reported condition or equipment.

Site-Manager-Approval TEXT 10

Identifies the ID number of the site manager that approved the IAR.

Time-Meter NUMERIC 14

Identifies the time meter reading of the JETDS item at the point of problem discovery.

NETWORK TROUBLE

PRIMARY KEYS: NTR-Number

FOREIGN KEYS:

Originator PERSON.ID-Number

ATTRIBUTES:

Name	Type	Length
Assigned Priority	NUMERIC	1

The relative significance assigned by the system administrator.

Brief-Description	TEXT	78

A brief description of the problem.

Closure-Date	DATE	8

Identifies the date the NTR was closed.

Computer	TEXT	3

Identifies the type of machine (PC/Mac) on which the problem occurred.

Description	TEXT	78

Originators description of the problem.

Fixed-Version	TEXT	4

Identifies the program release that fixes the problem.

NTR-Number	TEXT	8

The unique identifier of a network trouble report.

NTR-Status	TEXT	8

Identifies whether the NTR is open or closed.

Originator	TEXT	10

Identifies the ID number of the originator of the trouble report.

Related-NTR	TEXT	8

Other network trouble reports (NTRs) that have bearing on this NTR.

Suggested-Priority	NUMERIC	1

The relative significance assigned by the originator.

System-Administrator-Remarks	TEXT	78

The system administrator's remarks on the problem and its resolution.

Trouble-Code	NUMERIC	2

A two letter code that identifies the category of the problem.

Version	TEXT	4

ORGANIZATION

PRIMARY KEYS: UIC

FOREIGN KEYS: None

ATTRIBUTES:

Name	Type	Length
Acronym	TEXT	15

Identifies the organization in terms of its accepted acronym.

Command-Name	TEXT	80

Identifies the name of an organization.

UIC	NUMERIC	5

Unit Identification Code. Uniquely Identifies a command.

PERSON

PRIMARY KEYS: ID-Number

FOREIGN KEYS:

Organization ORGANIZATION.UIC

ATTRIBUTES:

Name	Type	Length
Building	TEXT	6

Identifies the building where the person is located.

Classified-Address	TEXT	67

Identifies the address of the facility where the person receives classified mail.

Code	TEXT	16

Identifies the departmental code of the person if appropriate.

Department	TEXT	12

Identifies the person's department.

FAX-Check-Number	NUMERIC	10

Identifies the phone number to call before or after sending a FAX.

FAX-Number	NUMERIC	10

Identifies the FAX number of the person.

First-Name	TEXT	20

Identifies the first name of the person.

ID-Number	TEXT	10

Uniquely identifies a person in the Tomahawk community.

Job-Title	TEXT	42

Identifies the job title of the person.

Last-Name	TEXT	20

Identifies the last name of the person.

Level	TEXT	3

Identifies the building level where the person is located.

Mail-Code	TEXT	10

Identifies the mail code for the location of the person.

Organization	TEXT	6

Identifies the UIC of the organization to which the person belongs.

Phone-Number	TEXT	10

Identifies the phone number of the person.

Rank	TEXT	17

Identifies the military rank of the person if appropriate.

Remarks	TEXT	78

Contains things about a person not identified in other fields.

Responsibilities	TEXT	27

Identifies the persons responsibilities.

Room	TEXT	10

Identifies the room in which the person is located.

Secure-Phone-Number	TEXT	10

Identifies the secure phone of the person.

Unclassified-Address	TEXT	67

Identifies the address of the facility where the person receives unclassified mail.

PROBLEM

PRIMARY KEYS: Problem-Number

FOREIGN KEYS:

Hull-Number	SHIP_SYSTEM.Hull-Number
JETDS	SHIP_SYSTEM.JETDS
Prepared-By	PERSON.ID-Number
Problem-Coordinator	PERSON.ID-Number
Test-Date	SHIP_TEST.Test-Date
Test-Name	SHIP_TEST.Test-Name

ATTRIBUTES:

Name	Type	Length
Action-Taken	TEXT	23

Used by the TDD to annotate the problem solution.

Date-Action-Taken	TEXT	8

Identifies the date that all local action on the problem was completed.

Date-Discovered	TEXT	8

Identifies the date that the problem was discovered.

Hull-Number	TEXT	6

Identifies the platform.

JETDS	TEXT	14

Identifies the equipment level nomenclature used to identify the equipment referenced.

Prepared-By	TEXT	10

Identifies the ID number of the author of the report.

Prepared-By-Date	TEXT	8

Identifies the original date for the problem report.

Problem-Coordinator	TEXT	10

Identifies the ID number of the individual coordinating the problem solution.

Problem-Description	TEXT	78

Describes the problem in free form remarks.

Problem-Number	TEXT	9

Identifies the platform and problem sequence number assigned to the problem.

Test-Date	TEXT	8

Identifies the date the test was performed.

Test-Name	TEXT	22

Identifies the type of test performed.

Test-Part-Step	TEXT	15

Identifies the portion of TWCS INCO test procedures that was being performed at the point of problem discovery.

Test-Procedure	TEXT	29

Identifies the test procedure that was being used at the point of problem discovery.

Test-Revision	TEXT	10

Identifies the revision of the test procedure that was being performed at the point of problem discovery.

SHIP

PRIMARY KEYS: Hull-Number

FOREIGN KEYS: None

ATTRIBUTES:

Name	Type	Length
Hull-Number	TEXT	6
Identifies the platform.		
Ship-Name	TEXT	22
Identifies the platform by name.		

SHIP_SYSTEM

PRIMARY KEYS: Hull-Number

JETDS

FOREIGN KEYS:

Hull-Number SHIP.Hull-Number

JETDS SYSTEM.JETDS

ATTRIBUTES:

Name	Type	Length
Hull-Number	TEXT	6

Identifies the platform.

JETDS	TEXT	14

Identifies the equipment level nomenclature used to identify the equipment referenced.

Serial-Number	TEXT	14

Identifies the fabrication serial number of the JETDS item.

SHIP_TEST

PRIMARY KEYS: Hull-Number

Test-Date

Test-Name

FOREIGN KEYS:

Hull-Number	SHIP.Hull-Number
Test-Name	TEST.Test-Name

ATTRIBUTES:

Name	Type	Length
Hull-Number	TEXT	6

Identifies the platform.

Period	TEXT	14

Identifies the ship's availability during which the test was conducted (ROH, PSA, REACT, CONST, OTHER).

Test-Date	TEXT	8

Identifies the date the test was performed.

Test-Name	TEXT	22

Identifies the type of test performed.

SYSTEM

PRIMARY KEYS: JETDS

FOREIGN KEYS: None.

ATTRIBUTES:

Name	Type	Length
JETDS	TEXT	14

Identifies the equipment level nomenclature used to identify the equipment referenced.

Noun-Name	TEXT	14

Contains the equipment level nomenclature used to identify the system referenced.

SYSTEM_UNIT

PRIMARY KEYS: JETDS

Unit-Number

FOREIGN KEYS:

JETDS SYSTEM.JETDS

Unit-Number UNIT.Unit-Number

ATTRIBUTES:

Name	Type	Length
JETDS	TEXT	14

Identifies the equipment level nomenclature used to identify the equipment referenced.

Unit-Number	TEXT	30

Identifies the unit equipment designator number.

TEST

PRIMARY KEYS: Test-Name

FOREIGN KEYS: None

ATTRIBUTES:

Name	Type	Length
Test-Description	TEXT	78

Describes the test performed.

Test-Name	TEXT	22

Identifies the type of test performed.

TEST_PROBLEM

PRIMARY KEYS: Problem-Number

FOREIGN KEYS:

Approved-TD	PERSON.ID-Number
Individual-Contacted	PERSON.ID-Number
Problem-Number	PROBLEM.Problem-Number

ATTRIBUTES:

Name	Type	Length
Approved-TD	TEXT	10

Identifies the ID number of the test director who approved the solution.

Date-Contacted	DATE	8

Identifies the date the individual was contacted about the problem.

Date-Needed	DATE	8

Identifies the date a solution is needed.

Date-TD	DATE	8

Identifies the date that the test director approved the final closure of the TPR.

Disposition	TEXT	1

Used by the test director to annotate that this is a recommended solution to the problem.

Individual-Contacted	TEXT	10

Identifies the ID number of the individual contacted by phone.

Passed-To-Shipyard	TEXT	1

Identifies whether the IAR/TPR was passed to the shipyard for action or information.

Problem-Impact	TEXT	78

Describes the impact on other hulls of the TPR.

Problem-Number	TEXT	9

Identifies the platform and problem sequence number assigned to the problem.

Problem-Type	TEXT	10

Describes the type of problem, whether PROCEDURE,EQUIPMENT, COMPUTER, SOFTWARE, or OTHER.

Solution	TEXT	1

Used to indicate whether the solution was an authorized interim solution, a final resolution, or a minor problem solution.

Solution-Approval	TEXT	1

Used by the TDD to annotate satisfactory or not acceptable solution to the problem.

TPR-File	TEXT	10

Identifies the file number assigned to the TPR by the LCSTDD.

TPR-Serial-Number	TEXT	10

Identifies the serial number assigned to the TPR by the LCSTDD..

UNIT

PRIMARY KEYS: Unit-Number

FOREIGN KEYS: None

ATTRIBUTES:

Name	Type	Length
Unit-Number	TEXT	30

Identifies the unit equipment designator number.

UNIT_CABLE

PRIMARY KEYS: Cable-Number

Unit-Number

FOREIGN KEYS:

Cable-Number UNIT_CABLE.Cable-Number

Unit-Number UNIT.Unit-Number

ATTRIBUTES:

Name	Type	Length
Cable-Number	TEXT	15

Identifies the cable designator number.

Unit-Number	TEXT	30

Identifies the unit equipment designator number.

APPENDIX D. TIMES ENTITY-RELATIONSHIP DIAGRAM

TIMES is a relational database designed using powerful CASE tools and a structured methodology. Several undocumented changes have been made since initial implementation. The E-R model of TIMES was reverse engineered to provide an accurate description of the database and a common basis for comparison with TEXN. The model produced is presented in the following pages.

An overview of the diagram is presented on the next page. This diagram is too small to read, thus a full size printout is also included. This printout takes nine pages and must be removed from this document to be viewed side by side. The diagram is divided into three columns, lettered A, B, and C, and three rows numbered 1, 2, and 3. Each page of the diagram has a column and row designation number on it (e.g., A1). Figure D-1 shows how the pages fit together. Placing the pages as shown will allow the diagram to be examined.

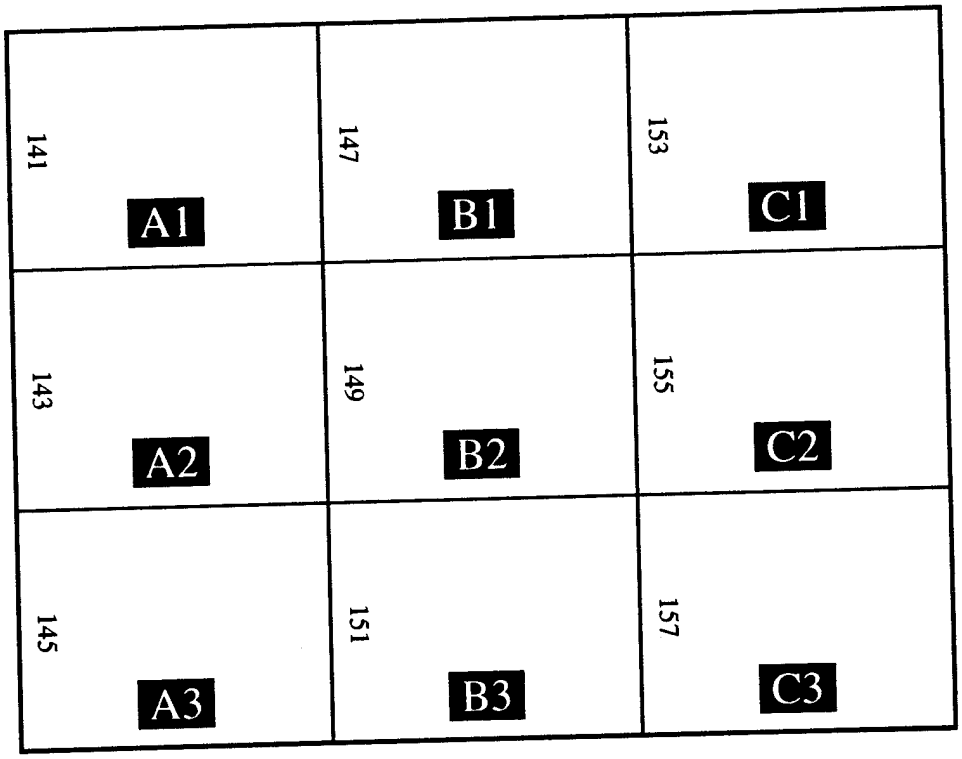
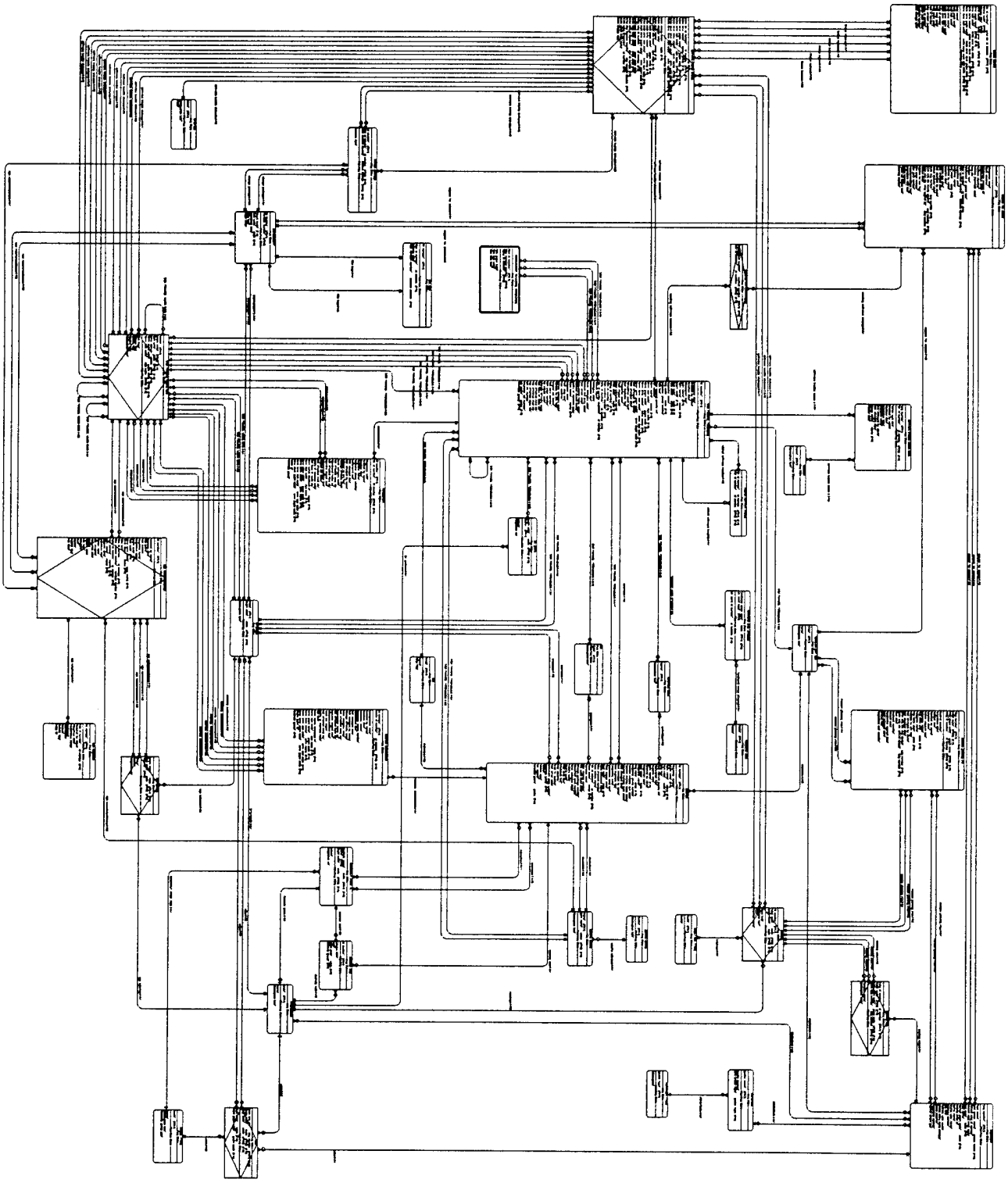
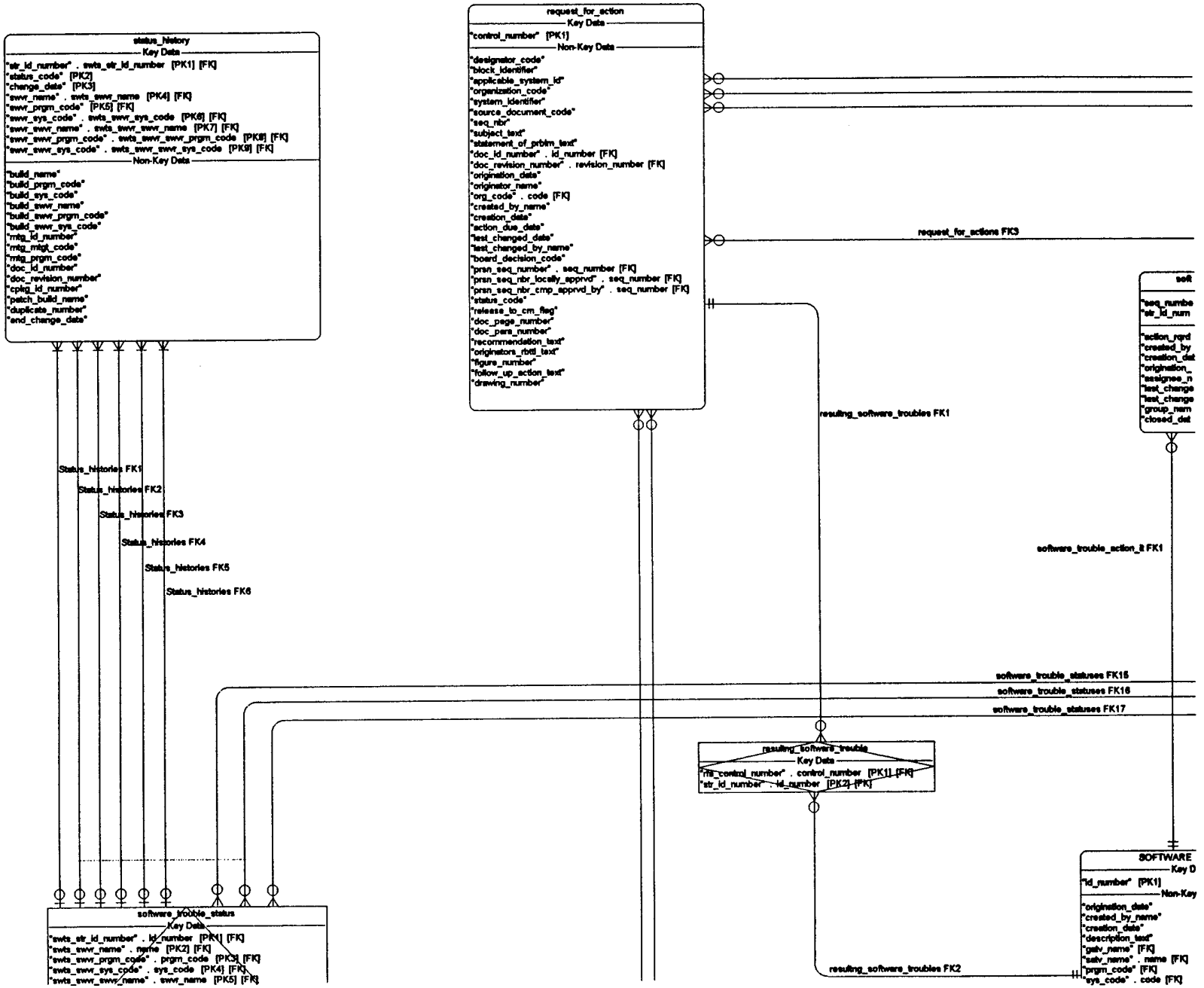
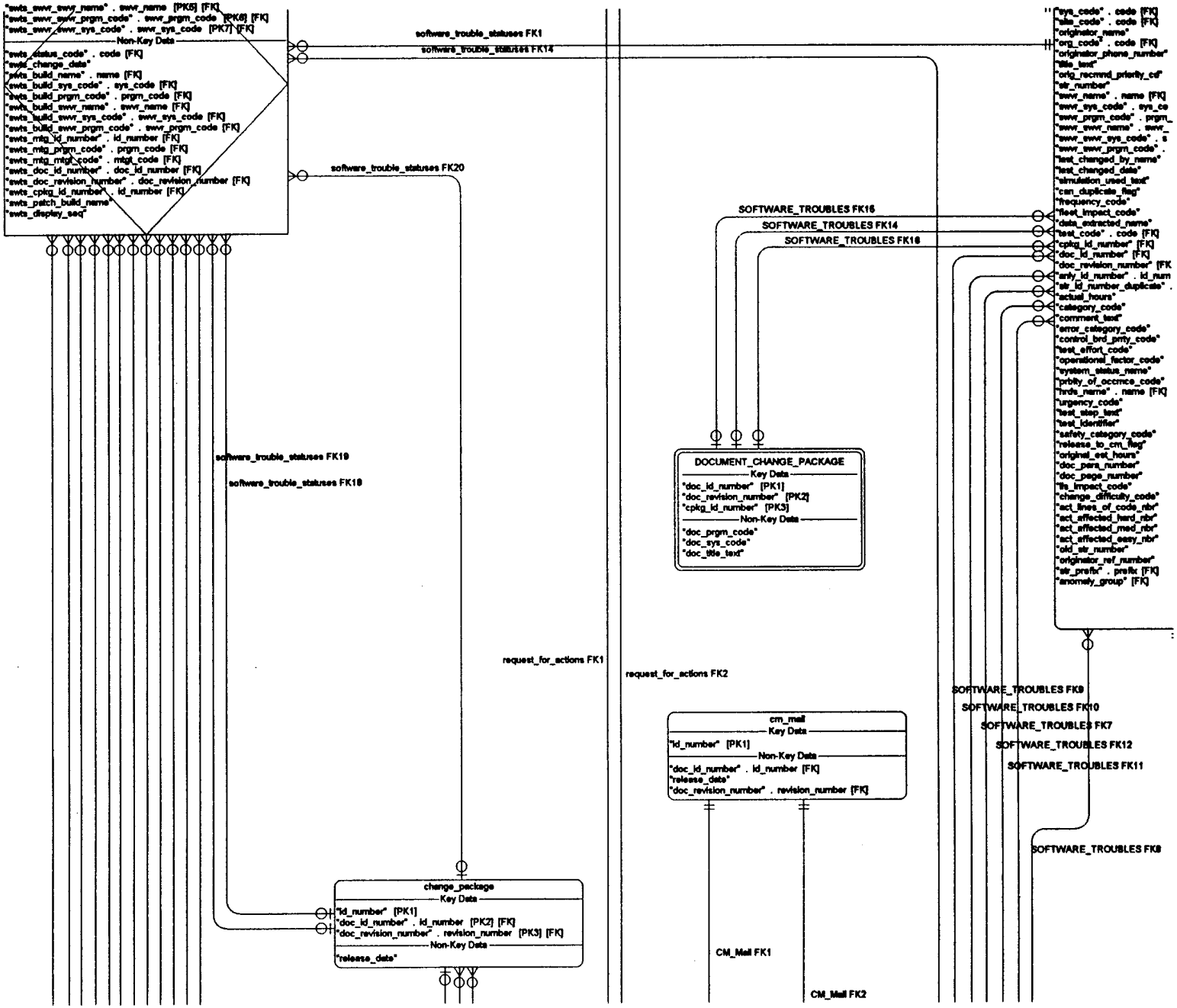


Figure D-1. TIMES E-R Diagram Map

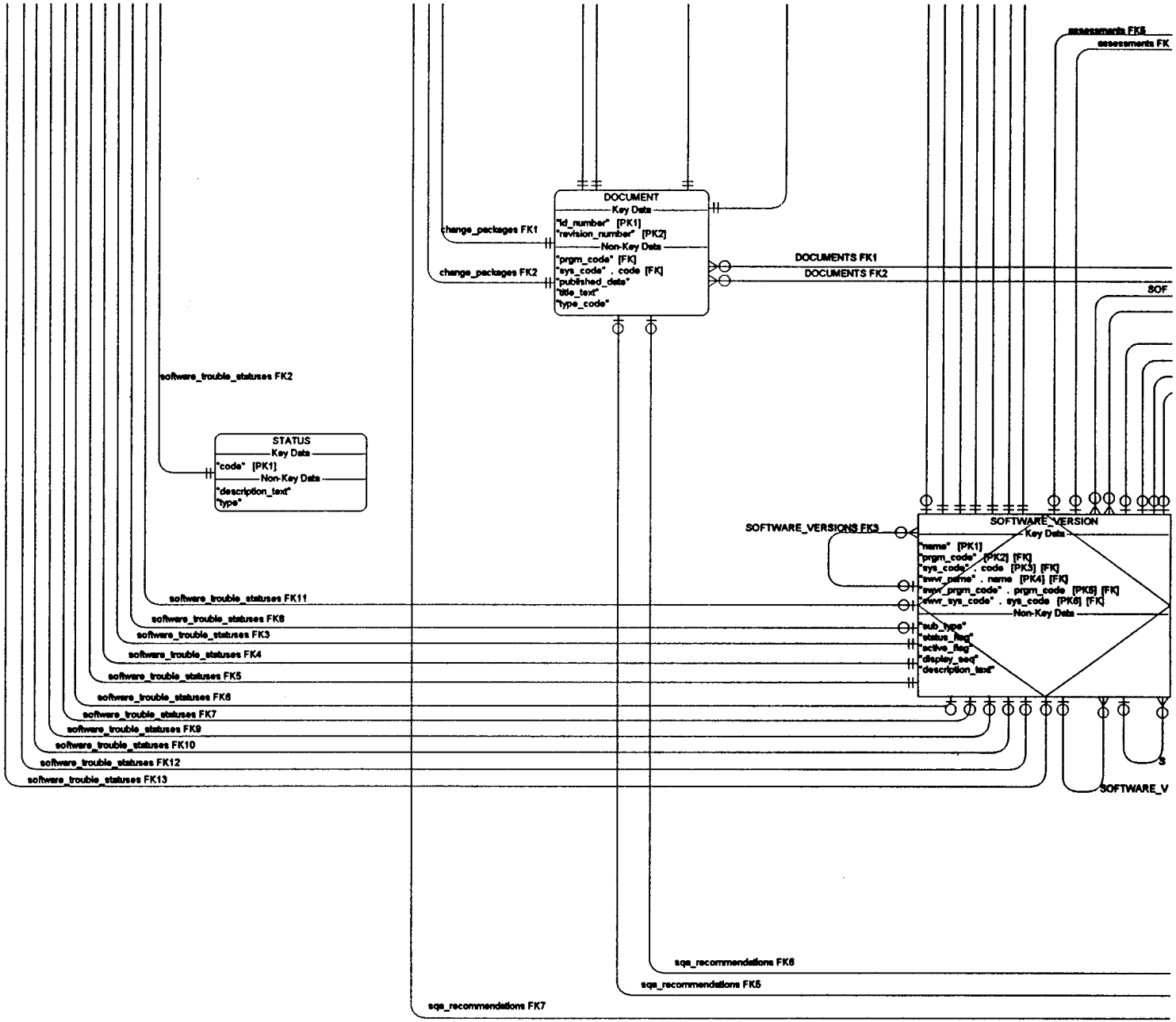



status_history
Key Data
"str_id_number" . swts_str_id_number [PK1] [FK]
"status_code" [PK2]
"change_date" [PK3]
"swvr_name" . swts_swvr_name [PK4] [FK]
"swvr_prgm_code" [PK5] [FK]
"swvr_sys_code" . swts_swvr_sys_code [PK6] [FK]
"swvr_swvr_name" . swts_swvr_swvr_name [PK7] [FK]
"swvr_swvr_prgm_code" . swts_swvr_swvr_prgm_code [PK8] [FK]
"swvr_swvr_sys_code" . swts_swvr_swvr_sys_code [PK9] [FK]
Non-Key Data
"build_name"
"build_prgm_code"
"build_sys_code"
"build_swvr_name"
"build_swvr_prgm_code"
"build_swvr_sys_code"
"rmtg_id_number"
"rmtg_mtgt_code"
"rmtg_prgm_code"
"doc_id_number"
"doc_revision_number"
"cptg_id_number"
"patch_build_name"
"duplicate_number"
"end_change_date"
request_for_action
Key Data
"control_number" [PK1]
Non-Key Data
"designator_code"
"block_identifier"
"applicable_system_id"
"organization_code"
"system_identifier"
"source_document_code"
"seq_nbr"
"subject_text"
"statement_of_prblm_text"
"doc_id_number" . id_number [FK]
"doc_revision_number" . revision_number [FK]
"origination_date"
"originator_name"
"org_code" . code [FK]
"created_by_name"
"creation_date"
"action_due_date"
"last_changed_date"
"last_changed_by_name"
"board_decision_code"
"prsn_seq_number" . seq_number [FK]
"prsn_seq_nbr_locally_apprvd" . seq_number [FK]
"prsn_seq_nbr_cmp_apprvd_by" . seq_number [FK]
"status_code"
"release_to_cm_flag"
"doc_page_number"
"doc_para_number"
"recommendation_text"
"originators_rbtl_text"
"figure_number"
"follow_up_action_text"
"drawing_number"
request_for_actions FK3
resulting_software_troubles FK1
Status_histories FK1
Status_histories FK2
Status_histories FK3
Status_histories FK4
Status_histories FK5
Status_histories FK6
software_trouble_action_lt FK1
software_trouble_statuses FK15
software_trouble_statuses FK16
software_trouble_statuses FK17
resulting_software_trouble
Key Data
"rst_control_number" . control_number [PK1] [FK]
"str_id_number" . id_number [PK2] [FK]
software_trouble_status
Key Data
"swts_str_id_number" . id_number [PK1] [FK]
"swts_swvr_name" . name [PK2] [FK]
"swts_swvr_prgm_code" . prgm_code [PK3] [FK]
"swts_swvr_sys_code" . sys_code [PK4] [FK]
"swts_swvr_swvr_name" . swvr_name [PK5] [FK]
resulting_software_troubles FK2
SOFTWARE
"id_number" [PK1]
"origination_date"
"created_by_name"
"creation_date"
"description_text"
"gsiv_name" [FK]
"ssiv_name" . name [FK]
"prgm_code" [FK]
"sys_code" . code [FK]

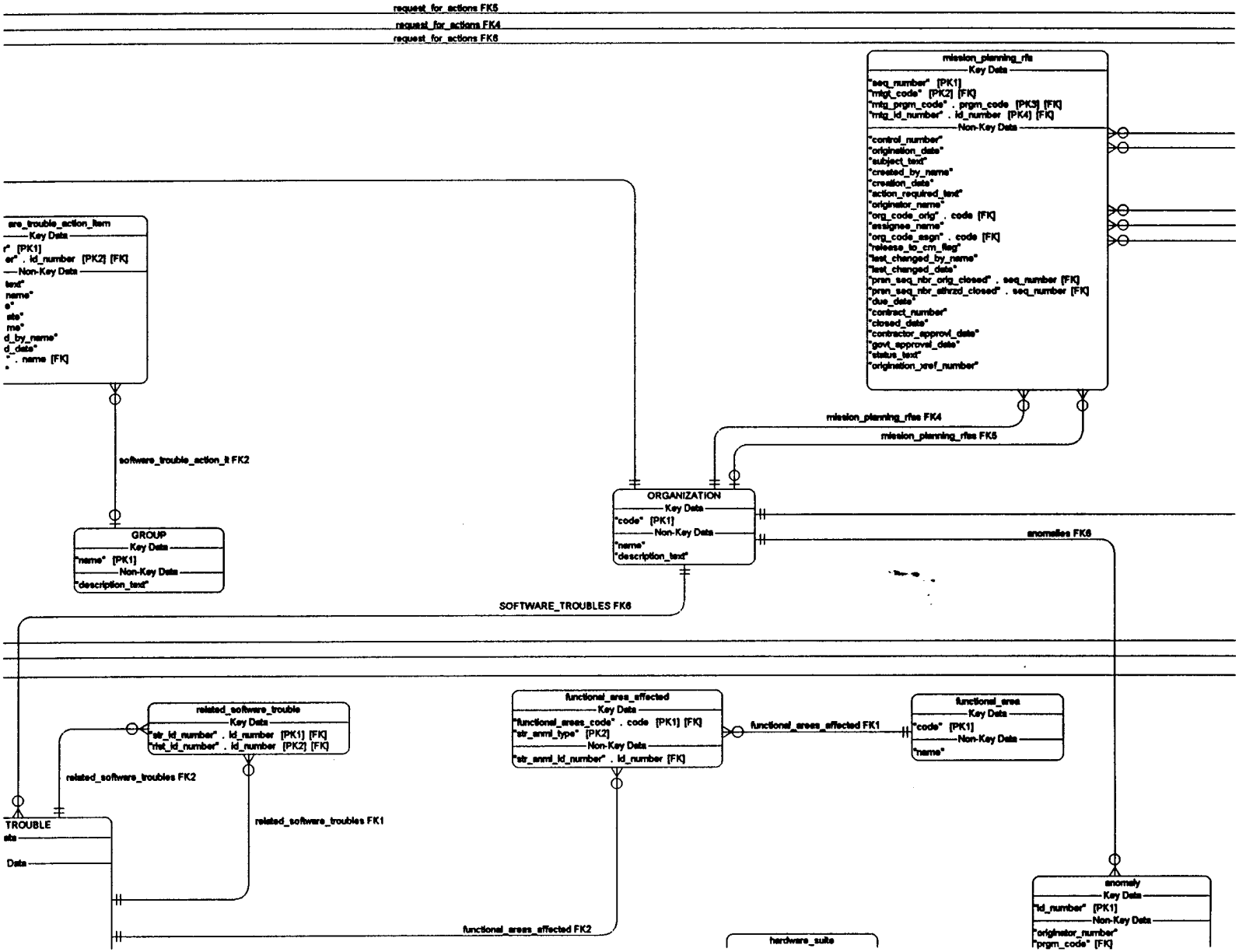
A1


"swts_swvr_swvr_name" . swvr_name [PK5] [FK]
"swts_swvr_swvr_prgm_code" . swvr_prgm_code [PK6] [FK]
"swts_swvr_swvr_sys_code" . swvr_sys_code [PK7] [FK]
Non-Key Data
"swts_status_code" . code [FK]
"swts_change_date"
"swts_build_name" . name [FK]
"swts_build_sys_code" . sys_code [FK]
"swts_build_prgm_code" . prgm_code [FK]
"swts_build_swvr_name" . swvr_name [FK]
"swts_build_swvr_sys_code" . swvr_sys_code [FK]
"swts_build_swvr_prgm_code" . swvr_prgm_code [FK]
"swts_mtg_id_number" . id_number [FK]
"swts_mtg_prgm_code" . prgm_code [FK]
"swts_mtg_mtgt_code" . mtgt_code [FK]
"swts_doc_id_number" . doc_id_number [FK]
"swts_doc_revision_number" . doc_revision_number [FK]
"swts_cpkg_id_number" . id_number [FK]
"swts_patch_build_name"
"swts_display_seq"
software_trouble_statuses FK1
software_trouble_statuses FK14
software_trouble_statuses FK20
software_trouble_statuses FK19
software_trouble_statuses FK18
request_for_actions FK1
request_for_actions FK2
SOFTWARE_TROUBLES FK16
SOFTWARE_TROUBLES FK14
SOFTWARE_TROUBLES FK18
DOCUMENT_CHANGE_PACKAGE
Key Data
"doc_id_number" [PK1]
"doc_revision_number" [PK2]
"cpkg_id_number" [PK3]
Non-Key Data
"doc_prgm_code"
"doc_sys_code"
"doc_title_text"
change_package
Key Data
"id_number" [PK1]
"doc_id_number" . id_number [PK2] [FK]
"doc_revision_number" . revision_number [PK3] [FK]
Non-Key Data
"release_date"
cm_mail
Key Data
"id_number" [PK1]
Non-Key Data
"doc_id_number" . id_number [FK]
"release_date"
"doc_revision_number" . revision_number [FK]
CM_Mail FK1
CM_Mail FK2
"sys_code" . code [FK]
"site_code" . code [FK]
"originator_name"
"org_code" . code [FK]
"originator_phone_number"
"title_text"
"orig_recmnd_priority_cd"
"str_number"
"swvr_name" . name [FK]
"swvr_sys_code" . sys_co
"swvr_prgm_code" . prgm_
"swvr_swvr_name" . swvr_
"swvr_swvr_sys_code" . s
"swvr_swvr_prgm_code" .
"test_changed_by_name"
"test_changed_date"
"simulation_used_text"
"can_duplicate_flag"
"frequency_code"
"fleet_impact_code"
"data_extracted_name"
"test_code" . code [FK]
"cpkg_id_number" [FK]
"doc_id_number" [FK]
"doc_revision_number" [FK
"anly_id_number" . id_num
"str_id_number_duplicate" .
"actual_hours"
"category_code"
"comment_text"
"error_category_code"
"control_brd_pnty_code"
"test_effort_code"
"operational_factor_code"
"system_status_name"
"prbty_of_occrnce_code"
"hrds_name" . name [FK]
"urgency_code"
"test_step_text"
"test_identifier"
"safety_category_code"
"release_to_cm_flag"
"original_est_hours"
"doc_para_number"
"doc_page_number"
"fls_impact_code"
"change_difficulty_code"
"act_lines_of_code_nbr"
"act_effected_hard_nbr"
"act_effected_med_nbr"
"act_effected_easy_nbr"
"old_str_number"
"originator_ref_number"
"str_prefix" . prefix [FK]
"anomaly_group" [FK]
SOFTWARE_TROUBLES FK9
SOFTWARE_TROUBLES FK10
SOFTWARE_TROUBLES FK7
SOFTWARE_TROUBLES FK12
SOFTWARE_TROUBLES FK11
SOFTWARE_TROUBLES FK8

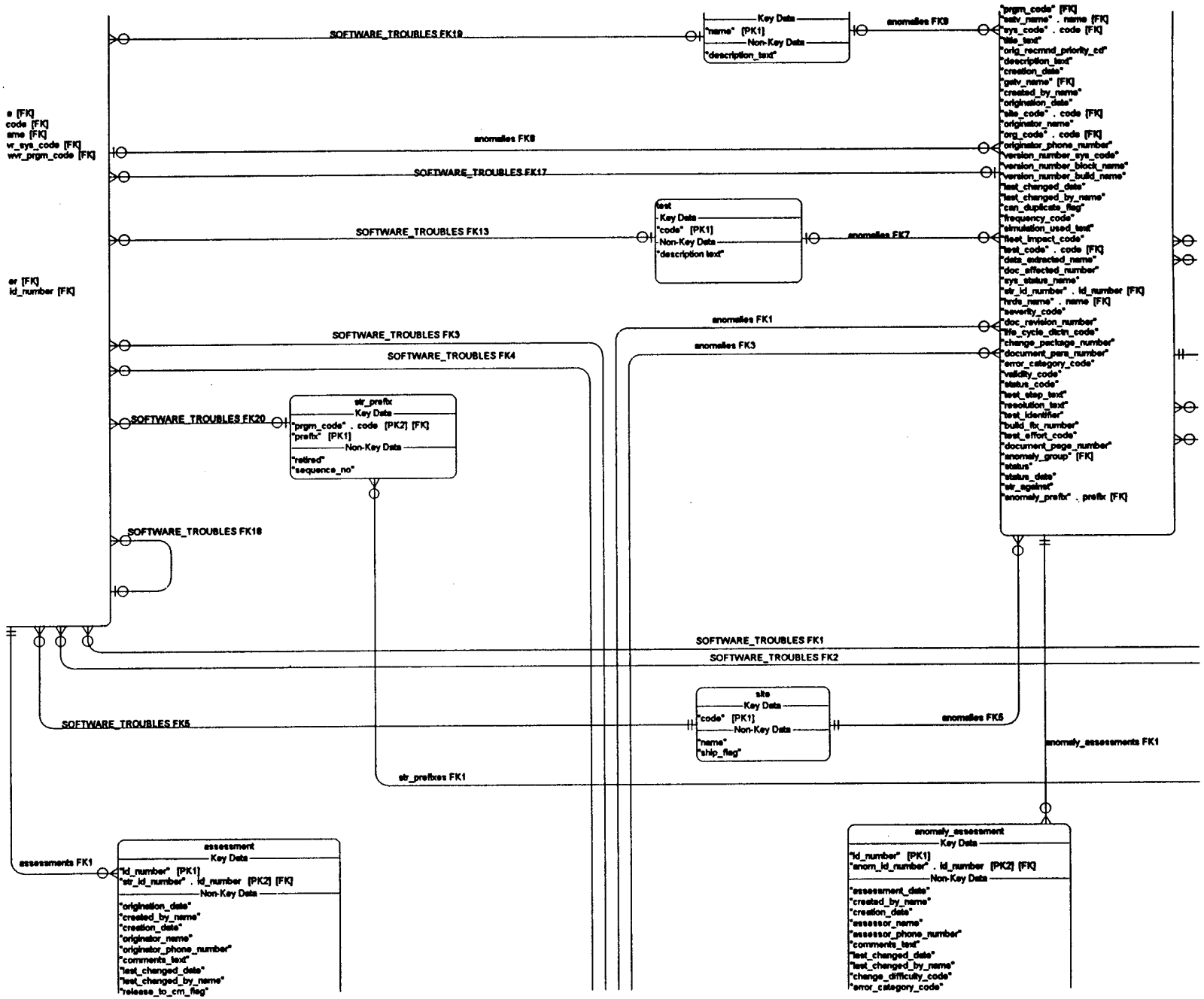
A2


DOCUMENT
Key Data
"id_number" [PK1]
"revision_number" [PK2]
Non-Key Data
"prgm_code" [FK]
"sys_code" . code [FK]
"published_date"
"title_text"
"type_code"
change_packages FK1
change_packages FK2
DOCUMENTS FK1
DOCUMENTS FK2
SOF
STATUS
Key Data
"code" [PK1]
Non-Key Data
"description_text"
"type"
software_trouble_statuses FK2
SOFTWARE_VERSIONS FK3
SOFTWARE_VERSION
Key Data
"name" [PK1]
"prgm_code" [PK2] [FK]
"sys_code" . code [PK3] [FK]
"swvr_name" . name [PK4] [FK]
"swvr_prgm_code" . prgm_code [PK5] [FK]
"swvr_sys_code" . sys_code [PK6] [FK]
Non-Key Data
"sub_type"
"status_flag"
"active_flag"
"display_seq"
"description_text"
software_trouble_statuses FK11
software_trouble_statuses FK8
software_trouble_statuses FK3
software_trouble_statuses FK4
software_trouble_statuses FK5
software_trouble_statuses FK6
software_trouble_statuses FK7
software_trouble_statuses FK9
software_trouble_statuses FK10
software_trouble_statuses FK12
software_trouble_statuses FK13
sqa_recommendations FK6
sqa_recommendations FK5
sqa_recommendations FK7

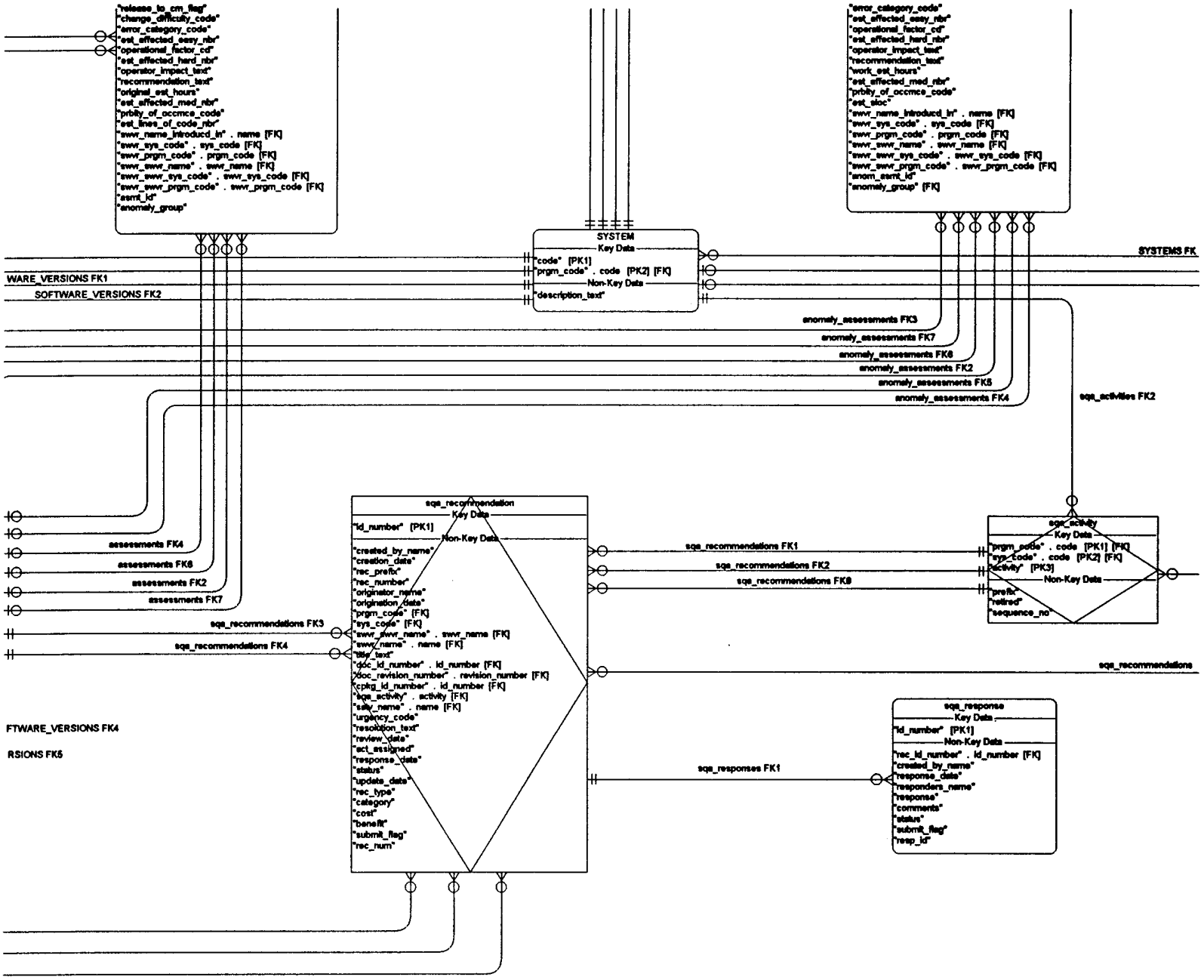
A3


request_for_actions FK5
request_for_actions FK4
request_for_actions FK6
mission_planning_rfs
Key Data
"seq_number" [PK1]
"mtgt_code" [PK2] [FK]
"mtg_prgm_code" . prgm_code [PK3] [FK]
"mtg_id_number" . id_number [PK4] [FK]
Non-Key Data
"control_number"
"origination_date"
"subject_text"
"created_by_name"
"creation_date"
"action_required_text"
"originator_name"
"org_code_orig" . code [FK]
"assignee_name"
"org_code_asgn" . code [FK]
"release_to_cm_flag"
"last_changed_by_name"
"last_changed_date"
"prsn_seq_nbr_orig_closed" . seq_number [FK]
"prsn_seq_nbr_athrzd_closed" . seq_number [FK]
"due_date"
"contract_number"
"closed_date"
"contractor_approval_date"
"govt_approval_date"
"status_text"
"origination_xref_number"
mission_planning_rfss FK4
mission_planning_rfss FK5
software_trouble_action_lt FK2
GROUP
Key Data
"name" [PK1]
Non-Key Data
"description_text"
ORGANIZATION
Key Data
"code" [PK1]
Non-Key Data
"name"
"description_text"
anomalies FK6
SOFTWARE_TROUBLES FK6
related_software_trouble
Key Data
"str_id_number" . id_number [PK1] [FK]
"rlst_id_number" . id_number [PK2] [FK]
related_software_troubles FK2
related_software_troubles FK1
functional_areas_affected
Key Data
"functional_areas_code" . code [PK1] [FK]
"str_anml_type" [PK2]
Non-Key Data
"str_anml_id_number" . id_number [FK]
functional_areas_affected FK1
functional_area
Key Data
"code" [PK1]
Non-Key Data
"name"
TROUBLE
Data
functional_areas_affected FK2
anomaly
Key Data
"id_number" [PK1]
Non-Key Data
"originator_number"
"prgm_code" [FK]
hardware_suite

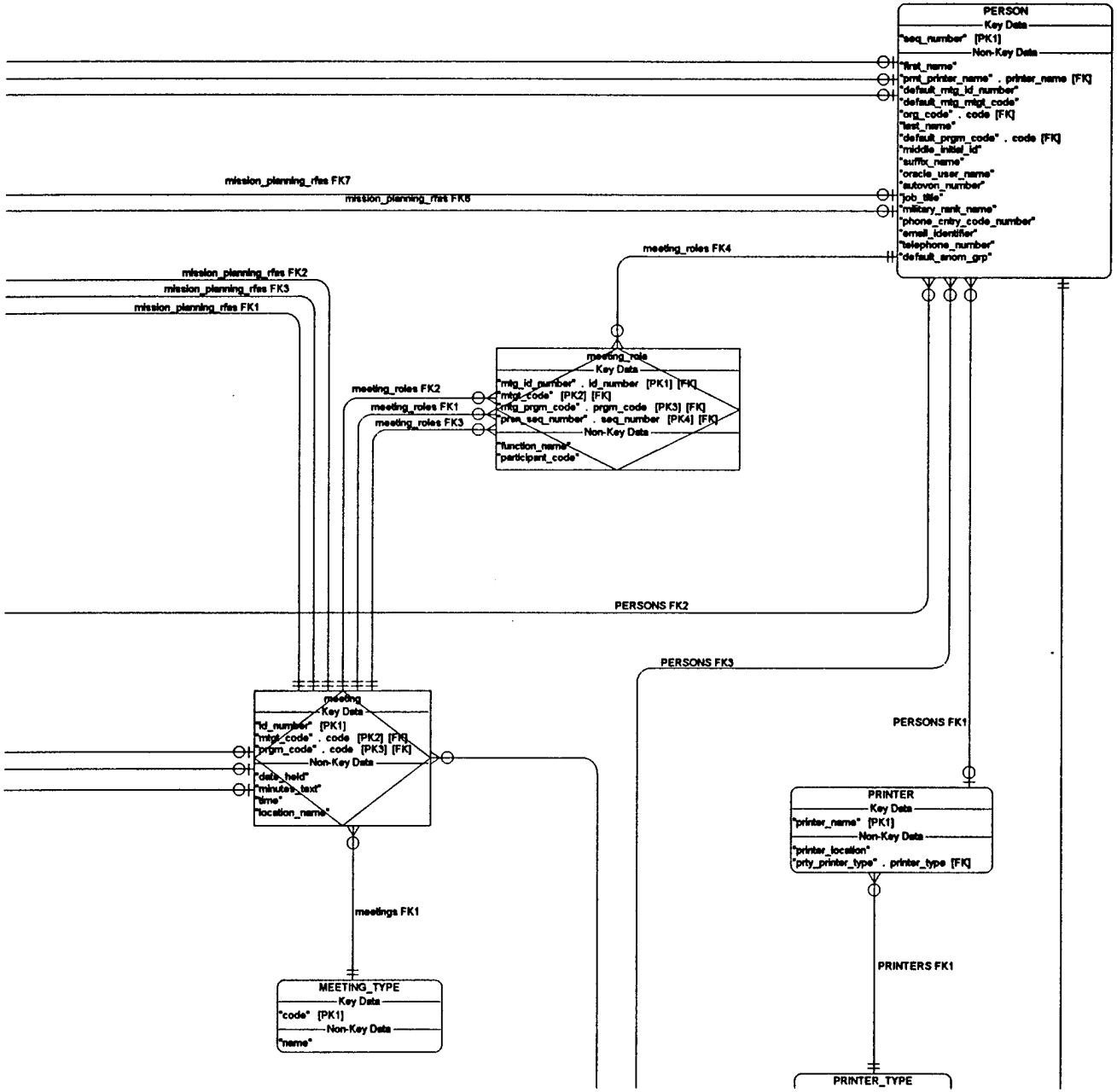
B1


SOFTWARE_TROUBLES FK19
Key Data
"name" [PK1]
Non-Key Data
"description_text"
anomalies FK8
"prgm_code" [FK]
"sstv_name" . name [FK]
"sys_code" . code [FK]
"title_text"
"orig_recmnd_priority_cd"
"description_text"
"creation_date"
"gstv_name" [FK]
"created_by_name"
"origination_date"
"site_code" . code [FK]
"originator_name"
"org_code" . code [FK]
"originator_phone_number"
"version_number_sys_code"
"version_number_block_name"
"version_number_build_name"
"last_changed_date"
"last_changed_by_name"
"can_duplicate_flag"
"frequency_code"
"simulation_used_text"
"fleet_impact_code"
"test_code" . code [FK]
"data_extracted_name"
"doc_affected_number"
"sys_status_name"
"str_id_number" . id_number [FK]
"trds_name" . name [FK]
"severity_code"
"doc_revision_number"
"life_cycle_dictn_code"
"change_package_number"
"document_para_number"
"error_category_code"
"validity_code"
"status_code"
"test_step_text"
"resolution_text"
"test_identifier"
"build_fix_number"
"test_effort_code"
"document_page_number"
"anomaly_group" [FK]
"status"
"status_date"
"str_against"
"anomaly_prefix" . prefix [FK]
e [FK]
code [FK]
ame [FK]
vr_sys_code [FK]
wvr_prgm_code [FK]
anomalies FK8
SOFTWARE_TROUBLES FK17
SOFTWARE_TROUBLES FK13
test
Key Data
"code" [PK1]
Non-Key Data
"description text"
anomalies FK7
er [FK]
id_number [FK]
SOFTWARE_TROUBLES FK3
SOFTWARE_TROUBLES FK4
anomalies FK1
anomalies FK3
SOFTWARE_TROUBLES FK20
str_prefix
Key Data
"prgm_code" . code [PK2] [FK]
"prefix" [PK1]
Non-Key Data
"retired"
"sequence_no"
SOFTWARE_TROUBLES FK18
SOFTWARE_TROUBLES FK1
SOFTWARE_TROUBLES FK2
site
Key Data
"code" [PK1]
Non-Key Data
"name"
"ship_flag"
SOFTWARE_TROUBLES FK5
anomalies FK5
anomaly_assessments FK1
str_prefixes FK1
assessments FK1
assessment
Key Data
"id_number" [PK1]
"str_id_number" . id_number [PK2] [FK]
Non-Key Data
"origination_date"
"created_by_name"
"creation_date"
"originator_name"
"originator_phone_number"
"comments_text"
"last_changed_date"
"last_changed_by_name"
"release_to_crn_flag"
anomaly_assessment
Key Data
"id_number" [PK1]
"anom_id_number" . id_number [PK2] [FK]
Non-Key Data
"assessment_date"
"created_by_name"
"creation_date"
"assessor_name"
"assessor_phone_number"
"comments_text"
"last_changed_date"
"last_changed_by_name"
"change_difficulty_code"
"error_category_code"

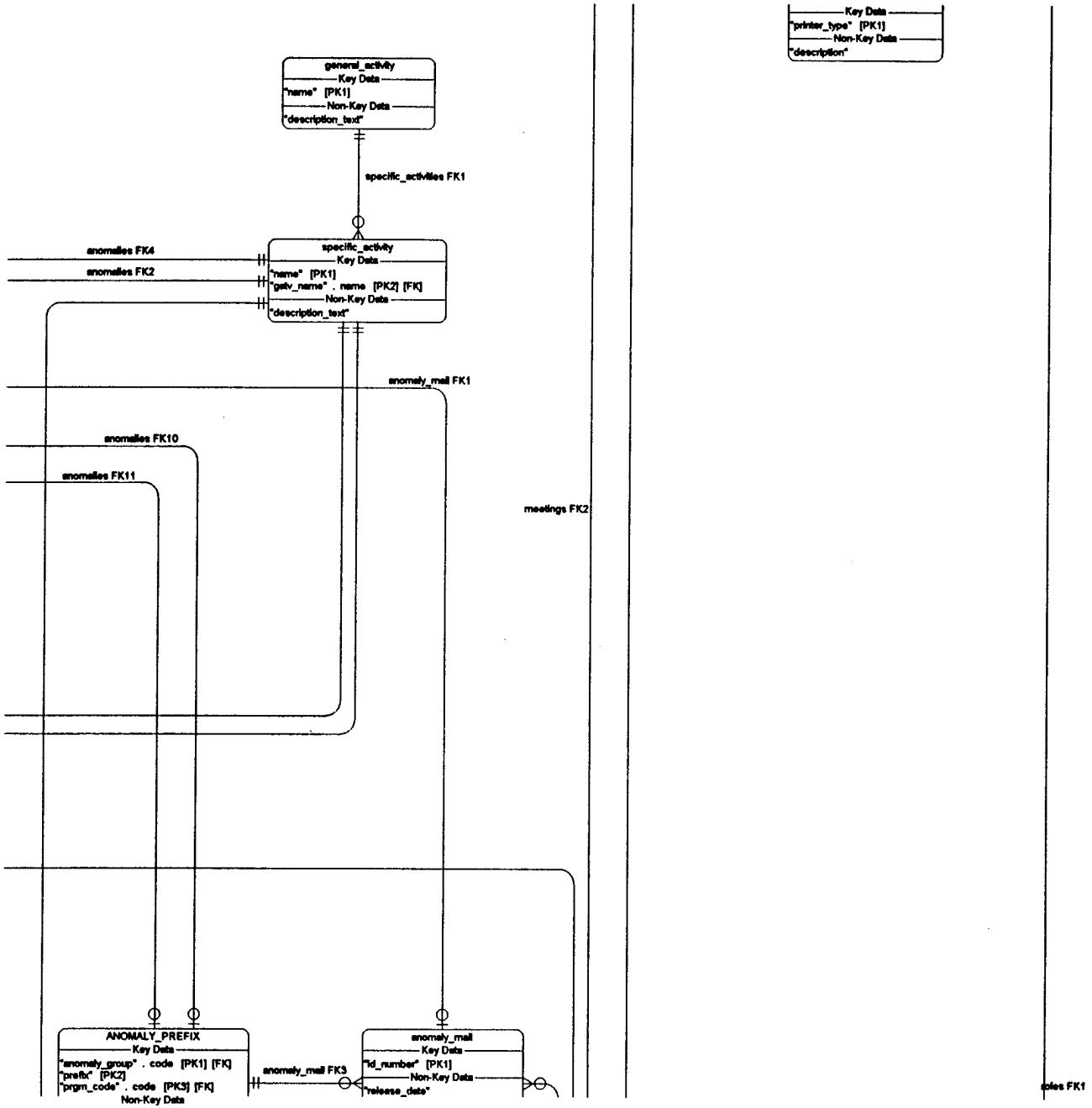
B2


"release_to_cm_flag"
"change_difficulty_code"
"error_category_code"
"est_affected_easy_nbr"
"operational_factor_cd"
"est_affected_hard_nbr"
"operator_impact_text"
"recommendation_text"
"original_est_hours"
"est_affected_med_nbr"
"prbblty_of_occrnce_code"
"est_lines_of_code_nbr"
"swvr_name_introducd_in" . name [FK]
"swvr_sys_code" . sys_code [FK]
"swvr_prgm_code" . prgm_code [FK]
"swvr_swvr_name" . swvr_name [FK]
"swvr_swvr_sys_code" . swvr_sys_code [FK]
"swvr_swvr_prgm_code" . swvr_prgm_code [FK]
"asmt_id"
"anomaly_group"
"error_category_code"
"est_affected_easy_nbr"
"operational_factor_cd"
"est_affected_hard_nbr"
"operator_impact_text"
"recommendation_text"
"work_est_hours"
"est_affected_med_nbr"
"prbblty_of_occrnce_code"
"est_sloc"
"swvr_name_introducd_in" . name [FK]
"swvr_sys_code" . sys_code [FK]
"swvr_prgm_code" . prgm_code [FK]
"swvr_swvr_name" . swvr_name [FK]
"swvr_swvr_sys_code" . swvr_sys_code [FK]
"swvr_swvr_prgm_code" . swvr_prgm_code [FK]
"anom_asmt_id"
"anomaly_group" [FK]
SYSTEM
Key Data
"code" [PK1]
"prgm_code" . code [PK2] [FK]
Non-Key Data
"description_text"
SYSTEMS FK
WARE_VERSIONS FK1
SOFTWARE_VERSIONS FK2
anomaly_assessments FK3
anomaly_assessments FK7
anomaly_assessments FK8
anomaly_assessments FK2
anomaly_assessments FK5
anomaly_assessments FK4
sqa_activities FK2
assessments FK4
assessments FK8
assessments FK2
assessments FK7
sqa_recommendations FK3
sqa_recommendations FK4
sqa_recommendation
Key Data
"id_number" [PK1]
Non-Key Data
"created_by_name"
"creation_date"
"rec_prefix"
"rec_number"
"originator_name"
"origination_date"
"prgm_code" [FK]
"sys_code" [FK]
"swvr_swvr_name" . swvr_name [FK]
"swvr_name" . name [FK]
"title_text"
"doc_id_number" . id_number [FK]
"doc_revision_number" . revision_number [FK]
"cpkg_id_number" . id_number [FK]
"sqa_activity" . activity [FK]
"sslv_name" . name [FK]
"urgency_code"
"resolution_text"
"review_date"
"act_assigned"
"response_date"
"status"
"update_date"
"rec_type"
"category"
"cost"
"benefit"
"submit_flag"
"rec_num"
sqa_recommendations FK1
sqa_recommendations FK2
sqa_recommendations FK8
sqa_activity
Key Data
"prgm_code" . code [PK1] [FK]
"sys_code" . code [PK2] [FK]
"activity" [PK3]
Non-Key Data
"prefix"
"retired"
"sequence_no"
sqa_recommendations
sqa_responses FK1
sqa_response
Key Data
"id_number" [PK1]
Non-Key Data
"rec_id_number" . id_number [FK]
"created_by_name"
"response_date"
"responders_name"
"response"
"comments"
"status"
"submit_flag"
"resp_id"
FTWARE_VERSIONS FK4
RSIONS FK5

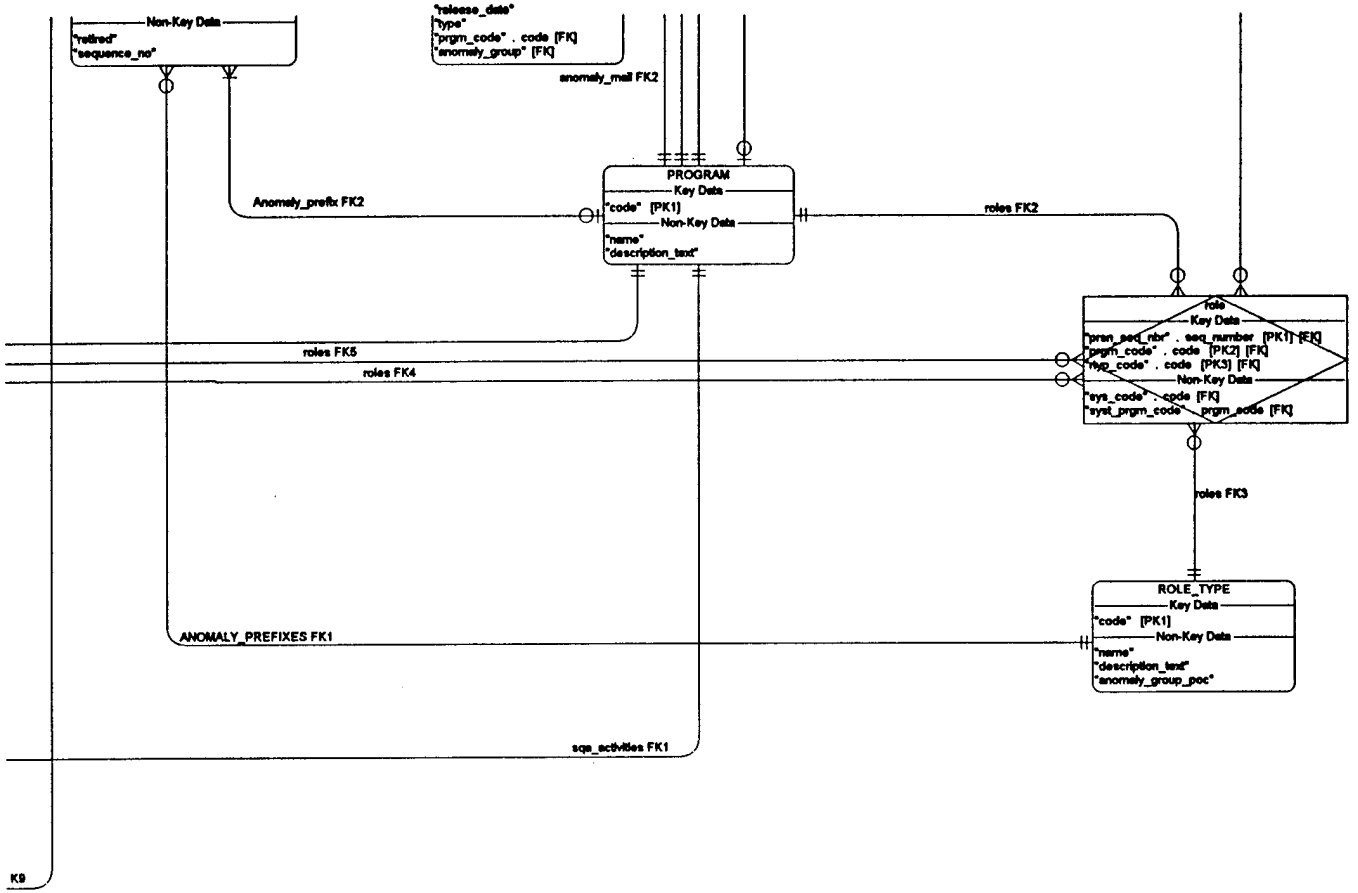
B3


PERSON
Key Data
"seq_number" [PK1]
Non-Key Data
"first_name"
"prnt_printer_name" . printer_name [FK]
"default_mtg_id_number"
"default_mtg_mtgt_code"
"org_code" . code [FK]
"last_name"
"default_prgm_code" . code [FK]
"middle_initial_id"
"suffix_name"
"oracle_user_name"
"autovon_number"
"job_title"
"military_rank_name"
"phone_cntry_code_number"
"email_identifier"
"telephone_number"
"default_anom_grp"
mission_planning_rfas FK7
mission_planning_rfas FK6
meeting_roles FK4
mission_planning_rfas FK2
mission_planning_rfas FK3
mission_planning_rfas FK1
meeting_role
Key Data
"mtg_id_number" . id_number [PK1] [FK]
"mtgt_code" [PK2] [FK]
"mtg_prgm_code" . prgm_code [PK3] [FK]
"prsn_seq_number" . seq_number [PK4] [FK]
Non-Key Data
"function_name"
"participant_code"
meeting_roles FK2
meeting_roles FK1
meeting_roles FK3
PERSONS FK2
PERSONS FK3
PERSONS FK1
meeting
Key Data
"id_number" [PK1]
"mtgt_code" . code [PK2] [FK]
"prgm_code" . code [PK3] [FK]
Non-Key Data
"date_held"
"minutes_text"
"time"
"location_name"
PRINTER
Key Data
"printer_name" [PK1]
Non-Key Data
"printer_location"
"prty_printer_type" . printer_type [FK]
meetings FK1
PRINTERS FK1
MEETING_TYPE
Key Data
"code" [PK1]
Non-Key Data
"name"
PRINTER_TYPE

C1


general_activity
Key Data
"name" [PK1]
Non-Key Data
"description_text"
specific_activities FK1
specific_activity
Key Data
"name" [PK1]
"gstv_name" . name [PK2] [FK]
Non-Key Data
"description_text"
anomalies FK4
anomalies FK2
anomaly_mail FK1
anomalies FK10
anomalies FK11
meetings FK2
Key Data
"printer_type" [PK1]
Non-Key Data
"description"
ANOMALY_PREFIX
Key Data
"anomaly_group" . code [PK1] [FK]
"prefix" [PK2]
"prgm_code" . code [PK3] [FK]
Non-Key Data
anomaly_mail FK3
anomaly_mail
Key Data
"id_number" [PK1]
Non-Key Data
"release_date"
oles FK1

C2


Non-Key Data
"retired"
"sequence_no"
"release_date"
"type"
"prgm_code" . code [FK]
"anomaly_group" [FK]
anomaly_mail FK2
PROGRAM
Key Data
"code" [PK1]
Non-Key Data
"name"
"description_text"
Anomaly_prefix FK2
roles FK2
Role
Key Data
"pran_seq_nbr" . seq_number [PK1] [FK]
"prgm_code" . code [PK2] [FK]
"rtyp_code" . code [PK3] [FK]
Non-Key Data
"sys_code" . code [FK]
"syst_prgm_code" . prgm_code [FK]
roles FK5
roles FK4
roles FK3
ROLE_TYPE
Key Data
"code" [PK1]
Non-Key Data
"name"
"description_text"
"anomaly_group_poc"
ANOMALY_PREFIXES FK1
sqa_activities FK1
K9

C3

LIST OF REFERENCES

Elmasri R., and Navanthe S., *Fundamentals of Database Systems*, The Benjamin/Cummings Publishing Company, Inc., 1989.

Emory, C. W., and Cooper, D. R., *Business Research Methods,* Richard D. Irwin, Inc., 1991.

Inmon, W. H., *"What Is a Data Warehouse?,"* (http:/waltz.ncsl.nist.gov/cait/datawarehouse.html) 1995.

Kamel, M., "Identifying, Classifying, and Resolving Semantic Conflicts in Distributed Heterogenous Databases: A Case Study," *Journal of Database Management,* Winter 1995.

Kamel, M., and Ceruti, M., "Semantic Heterogeneity in Database and Data DIctionary Integration for Command and Control Systems," *Conference Proceedings, DoD Database Colloquium '94*, AFCEA, 1994.

NSWC DD, *TIMES Build Report,* NSWC Dahlgren Division,March 1993.

NSWC DD, *TIMES Data Dictionary Reference Manual,* NSWC Dahlgren Division,July 1993.

NSWC DD, *TIMES Design Report,* NSWC Dahlgren Division, April 1993.

NSWC DD, *Using TIMES,* NSWC Dahlgren Division,July 1993.

NSWC PHD, *"Engineering 2000 Overview,"* Slide Presentation, February 1995.

Orfali, R., Harkey, D., and Edwards, J., *Essential Client/Server Survival Guide,* John Wiley & Sons, Inc., 1994.

SAIC, *Engineering 2000 Feasibility Study,* SAIC, November 1994.

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center 2
   Cameron Station
   Alexandria, Virginia 22314-6145

2. Library, Code 013 2
   Naval Postgraduate School
   Monterey, California 93943-5101

3. Professor Magdi N. Kamel 2
   Department of Systems Management (Code SM/Ka)
   Naval Postgraduate School
   Monterey, CA 93943-5000

4. Professor Martin J. McCaffrey 1
   Department of Systems Management (Code SM/Mf)
   Naval Postgraduate School
   Monterey, CA 93943-5000

5. Cary F. Martinez, Code 4G53 2
   Port Hueneme Division, Naval Surface Warfare Center
   Tomahawk Systems Engineering
   4363 Missile Way
   Port Hueneme, CA 93043-4307

6. LT Thomas E. O'Neill, USN 1
   c/o Thomas E. O'Neill
   32331 N. River Rd.
   Mt. Clemens, MI 48045

7. LT Milton J. Prell, USN 1
   c/o John Prell
   3000 Long Grove Ct.
   Aurora, IL 60504